Algorithms for Polynomial and Rational Approximation

Ricardo Pachón

Exeter College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity Term 2010

a mi hermana

Acknowledgements

I am deeply grateful to Nick Trefethen, an inspiring supervisor and a mentor since I arrived to Oxford. It has been a privilege to be his student and I am indebted to him for his advice and encouragement through this project.

I had the pleasure of working very closely with postdoctoral researchers Rodrigo Platte, Pedro Gonnet and Joris van Deun. My sincere thanks go to the three of them for sharing their knowledge and contributing to my own research. Together with Ásgeir Birkisson, Toby Driscoll, Nick Hale and Mark Richardson, they were my colleagues in the Chebfun Team, of which I am proud to have been a member. My gratefulness goes to all of them.

I offer my thanks and enduring gratitude to everyone in the Numerical Analysis Group at Oxford. In particular I would like to acknowledge Nick Gould, Raphael Hauser, Endre Süli and Andy Wathen, and my friends and colleagues Siobhan Burke, Leonardo Figueroa, Jari Fowkes, Bernhard Langwallner, Tyrone Rees, Thomas Schmelzer and Martin Stoll. A special mention goes to Lotti Ekert, for her invaluable help on a day-to-day basis.

My gratitude goes to Jean Paul Berrut and Ian Sobey, the examiners of this thesis, who provided me with thorough comments and suggestions to improve the original manuscript. Needless to say, any errors or oversights are mine alone.

I was one of the founders of the University of Oxford SIAM Student Chapter and a member of the committee for two years. This was a tremendously enriching experience and I would like to acknowledge David Hewett, David Knezevic, Sarah McBurnie, Mason Porter and Thomas Woolley.

My sincere gratitude to all my friends at Oxford, many at Exeter College, for making the last four years a unique and vibrant experience.

I would like to acknowlege the funding by the Engineering and Physical Sciences Research Council, Oxford University Computing Laboratory, Universidad Católica de Colombia, and Exeter College.

Finally, all my love goes to my father, my mother and my sister, who have been supporting me at every step since I left Colombia six years ago.

Abstract

Robust algorithms for the approximation of functions are studied and developed in this thesis. Novel results and algorithms on piecewise polynomial interpolation, rational interpolation and best polynomial and rational approximations are presented.

Algorithms for the extension of Chebfun, a software system for the numerical computation with functions, are described. These algorithms allow the construction and manipulation of piecewise smooth functions numerically with machine precision. Breakpoints delimiting subintervals are introduced explicitly, implicitly or automatically, the latter method combining recursive subdivision and edge detection techniques.

For interpolation by rational functions with free poles, a novel method is presented. When the interpolation nodes are roots of unity or Chebyshev points the algorithm is particularly simple and relies on discrete Fourier transform matrices, which results in a fast implementation using the Fast Fourier Transform. The method is generalised for arbitrary grids, which requires the construction of polynomials orthogonal on the set of interpolation nodes. The new algorithm has connections with other methods, particularly the work of Jacobi and Kronecker, Berrut and Mittelmann, and Eğecioğlu and Koç.

Computed rational interpolants are compared with the behaviour expected from the theory of convergence of these approximants, and the difficulties due to truncated arithmetic are explained. The appearance of common factors in the numerator and denominator due to finite precision arithmetic is characterised by the behaviour of the singular values of the linear system associated with the rational interpolation problem.

Finally, new Remez algorithms for the computation of best polynomial and rational approximations are presented. These algorithms rely on interpolation, for the computation of trial functions, and on Chebfun, for the location of trial references. For polynomials, the algorithm is particularly robust and efficient, and we report experiments with degrees in the thousands. For rational functions, we clarify the numerical issues that affect its application.

Contents

List of Figures

CHAPTER 1

Introduction

Although this may seem a paradox, all exact science is dominated by the idea of approximation. When a man tells you that he knows the exact truth about anything, you are safe in inferring that he is an inexact man.

Bertrand Russell
The Scientific Outlook (1931)

In 1934, Evgeny Yakovlevich Remez published in a series of three papers the algorithm that now bears his name for the computation of the *best polynomial approximation*, that is, the polynomial among those of a fixed degree that minimises the deviation in the supremum norm from a given continuous function on a given interval. We learn from Remez himself, in a footnote in one of these publications, that the first users of his algorithm were three female students at the University of Kiev—*Mesdemoiselles* Linnik, Kouniavska, and Masanovska. This revelation is not surprising, coming from a time when the role of women in mathematics was largely confined to that of a human computer, workers with the heavy labour of performing long and tedious numerical calculations for scientific purposes. The three women obtained the coefficients and errors of the best approximations to $|x|$ by polynomials of degrees 5, 7, 9, and 11, and though we do not know how tiresome this computation was, we know it was carried out meticulously, yielding all the figures accurate to about 4 decimal places.

Seventy-five years after these numerical experiments were performed, how large can we make the degree when computing best polynomial approximations? The advent of computers in the mid-20th century ignited interest in the Remez algorithm and we find numerous publications about it in the late 1950s and early 1960s, reporting examples with degrees of a couple of dozens. However, after this burst of investigation, research on the computation of best approximations diminished, perhaps in part due to the paucity of applications aside from digital filter design, a subject where it found its niche. The Remez

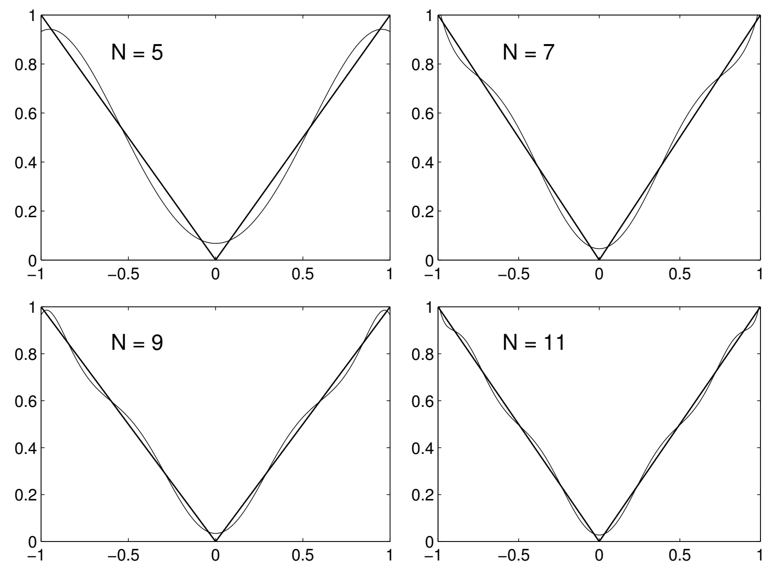

Figure 1.1: Best polynomial approximation (thin lines) of degrees 5, 7, 9, and 11 to $|x|$ on $[-1, 1]$. The coefficients and errors of these polynomials, accurate to 4 decimal places, were computed for the first time by Linnik, Kouniavska, and Masanovska, and published by Remez in 1934. The methods presented in this thesis allow one to compute best approximants to numerous functions with degrees in the thousands, typically accurate to 12–13 decimal places, in a matter of seconds.

algorithm became a mandatory topic in books of approximation theory but distant from practical usage, and the exploration of best polynomials with degrees in the hundreds or thousands never became a priority.

At the end of this thesis, in Chapter 6, we will present a new version of the Remez algorithm that makes it possible to compute best approximations with a single Chebfun command. For example, here we find the degree 100 best approximation to $\exp(|x|)$ in less than a second:

```
>> f = chebfun(@(x) exp(abs(x)),'splitting','on');
>> tic, p = remez(f,100); toc
Elapsed time is 0.307301 seconds.
>> norm(f-p,inf)
ans = 0.0028014408940777
```

With one further command, `plot(f-p)`, we obtain a curve with 103 points of equioscillation. The simplicity and speed of these computations changes the nature of the exploration of the Remez algorithm, making it an easy matter to quickly compare best approximants of many degrees, or for various different functions. See Figures 1.1 and 1.2.

This new way of computing best polynomial approximations is a good example of the main theme in this thesis: The study and development of robust algorithms for the approximation of functions. In the rest of this chapter we give a panoramic view of the main

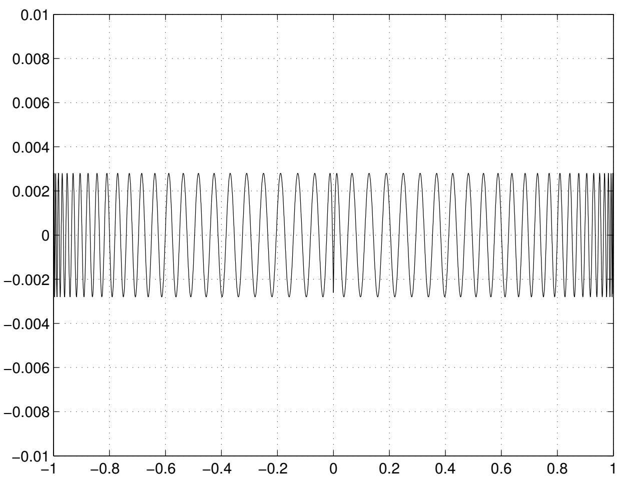

Figure 1.2: Error of the best polynomial approximation of degree 100 to the function $\exp(|x|)$ on $[-1, 1]$, computed with the methods of Chapter 6.

themes and relationships in this thesis, namely four types of approximations (interpolation, piecewise interpolation, approximations of maximum contact, and best approximation) and two families of approximating functions (polynomials and rational functions). We will also specify the main contributions of this thesis. The following table summarises which of these approximations are presented and in which areas we present novel results.

	Polynomials	Rational functions
Interpolation	presentation	**contribution**
Piecewise interpolation	**contribution**	—
Maximum contact	presentation	presentation
Best approximation	**contribution**	**contribution**

Structure of the thesis

Figure 1.3 shows a schematic representation of the structure of the thesis. Our point of departure is the summary of polynomial interpolation presented in Chapter 2. We review aspects of the construction and convergence of polynomial interpolants, with a special emphasis on interpolation in grids of roots of unity and Chebyshev points. Everything in this chapter, except its presentation and some isolated results of our own, is taken from the literature and should not be considered original. On the contrary, Chapter 2 brings together results that are scattered over more than a century and which have inspired our interest in approximation theory. To animate the discussion of this theoretical framework, we have selected six theorems of the literature for which we present complete proofs (Theorems 2.4–2.6 and 2.8–2.10).

Theory gives way to practice in Chapter 3, where our discussion revolves around the development of the *constructor methods* for piecewise polynomial approximation in Chebfun,

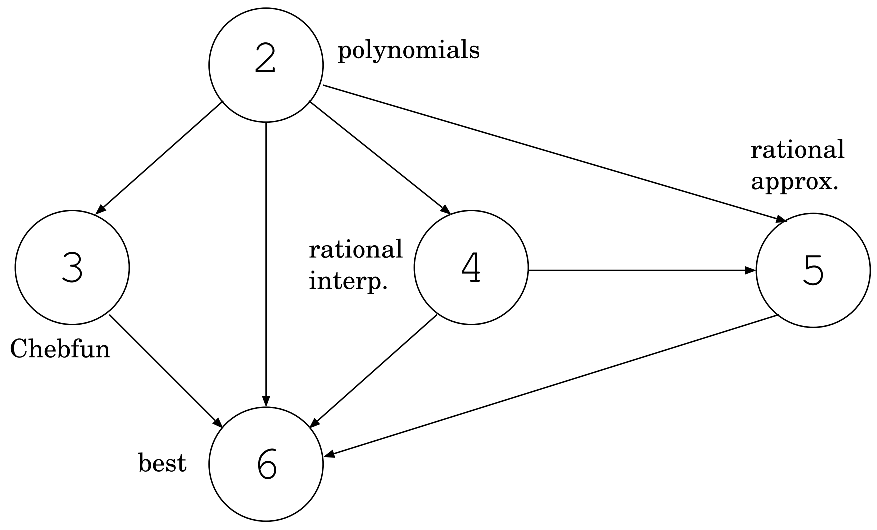

Figure 1.3: Structure of the thesis: The discussion of polynomial interpolation sets the foundation for the rest of thesis and the construction of the Remez algorithms combines tools presented throughout the document.

a software system at the core of our research. The theoretical background of this algorithm, as well as the motivation for looking for alternatives to polynomial interpolation, come directly from Chapter 2.

Another alternative to polynomial interpolation is rational interpolation. In Chapter 4 we study the *algebraic* aspect of the rational interpolation problem, that is, the construction of the numerator and denominator polynomials of a rational function that fits certain data. The key observation will be that the original nonlinear problem can be formulated in terms of an associated linear system. Similarly as in the polynomial case, rational interpolation in roots of unity and Chebyshev points offers a simple method for its computation. In this thesis we work exclusively with rational interpolants with free poles, not to be confused with the important category of rational interpolants with prescribed poles.

The capability of rational interpolation for approximation, what is often called the *analytic* aspect, is explored in Chapter 5. The first part, concerned with a description of the theoretical properties, follows closely the results presented previously for polynomial interpolants in Section 2.2. To stress this relation between polynomial and rational interpolation we present the proof of another classical theorem (Theorem 5.2) which resembles the techniques employed in those of Chapter 2. In the second part we move to study the properties of the *computed* rational interpolants and show that many elements interact in the numerical computation of rational interpolants. This can cause the computed approximant to differ greatly from the one expected from the theory.

Best approximation is a challenging problem, especially in the rational case. As can be inferred from the diagram in Figure 1.3, we employ material from the whole thesis for its solution. We need to emphasise that the rational Remez algorithm presented in Chapter 6 is not completely successful and does not have the robustness of its polynomial counterpart. Nevertheless, we believe that the progress made in this thesis towards the understanding of this problem is the most important in a long time, and not only indicates clearly where the fundamental difficulties are but also guides directions for its improvement. It is reasonable to think that the rational Remez algorithm presented here requires only a couple of new additional ideas to make it as robust as we desire.

The name "approximations of maximum contact" is not standard but groups what are known as truncations (for polynomials) and Padé-type approximants (for rational functions), both constructed from the first terms in the representation of a function as an infinite series. The former is presented in Section 2.3 and the later in Section 5.3. In particular we use the Chebyshev–Padé approximant for the initialisation of our rational Remez code.

Part of the action in this thesis happens in the complex plane. Some of the convergence properties for both polynomial and rational interpolants are explained by the capability of the underlying function to be analytically continued to a larger region in the complex domain. The construction of the interpolants can be made from grids in the complex plane and for both the polynomial and the rational case we emphasise this possibility. Additionally, our analysis of the convergence properties of the interpolants involves their behaviour in the complex domain, which can be used as a method for the numerical computation of the analytic continuation of a function or for the location of its singularities. Results in this direction are presented in Section 2.4 and 5.1.

At the end of each chapter we present a discussion and indications for future research, and in Chapter 7 we collect the main results of the thesis.

Contributions of the thesis

1. *New algorithm for the automatic construction of piecewise polynomial approximants* (see Section 3.3, pp. 43–48). Joint work with Rodrigo Platte and Lloyd N. Trefethen.

   This algorithm combines edge detection and recursive subdivision for the approximation of discontinuous functions and functions that cannot be accurately represented by a polynomial of a moderately high degree. Since this algorithm is at the heart of Chebfun, it is arguably the most successful, in terms of users, among the ones presented in this thesis.

   This algorithm is presented in R. Pachón, R. Platte, and L. N. Trefethen, *Piecewise smooth chebfuns*, IMA J. Numer. Anal. (in press, published online on July 2009).

2. *New algorithm for rational interpolation with free poles* (see Chapter 4 and in particular Theorem 4.1, p. 57). Joint work with Pedro Gonnet and Joris van Deun.

Perhaps the most important contribution in this thesis is the new algorithm for rational interpolation with free poles. For interpolation in roots of unity (Figure 4.1, p. 55) or Chebyshev points (Figures 4.2 and 4.3 on pp. 58 and 60), it is particularly efficient since it exploits the structure of the associated linear problem to solve it by the Fast Fourier Transform algorithm. At a theoretical level, it is closely related to orthogonal polynomials.

This algorithm is presented in R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., submitted.

3. *Characterisation of the discrepancy between exact and computed rational interpolants* (see Section 5.2, pp. 77–83).

   The behaviour of rational interpolants computed in floating-point arithmetic is usually not as expected. For purposes of approximation this difference is noticeable (see Figure 5.6, p. 75). This is the result of various entangled effects, some of which have not been previously discussed in the literature. In this thesis we settle many of these issues by providing a concrete characterisation of the effect of rounding errors on the computed rational interpolant (Figures 5.9 and 5.11 on pp. 80 and 82).

   Part of this contribution is presented in R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., submitted.

4. *New barycentric-Remez algorithm for best polynomial approximation* (see Sections 6.2, 6.3, and 6.4, pp. 98–113). Joint work with Lloyd N. Trefethen.

   In this thesis we propose a version of the Remez algorithm that makes it robust enough to efficiently compute to high accuracy polynomials of best approximation with degrees in the thousands. The algorithm requires a robust method for computing the barycentric weights (Figure 2.3, p. 19), an analytic formula for the levelled error (formula (6.13), p. 102), and the efficient Chebfun rootfinder (Section 3.4, p. 48). This contribution comes in the form of a "Ten Digit Algorithm" (see Figure 6.6, p. 105) and a thorough study of its properties.

   This algorithm is presented in R. Pachón and L. N. Trefethen, *Barycentric-Remez algorithms for best polynomial approximation in the chebfun system*, BIT Numer. Math., (2009) 49: 721–741.

5. *New interpolating-Remez algorithm for best rational approximation* (see Sections 6.2, 6.3, and 6.5, pp. 98–103 and 113–118).

   The 37-line code presented in Figure 6.13, p. 114, brings together algorithms and tools discussed throughout the document: Chebfun, rational interpolation, the Remez algorithm and Chebyshev–Padé approximation. Moreover, we can relate directly its behaviour to that of rational interpolants since it uses these as its building blocks. Apart from the new algorithm, our most important contribution regarding best rational approximation is to clarify the numerical issues that affect its computation.

The title of this thesis echoes those of the papers by Korevaar in 1980 and Saff in 1986, both called *Polynomial and rational approximation in the complex domain.* The distinctive word in ours, however, is *Algorithms.* In our research, we have found that the literature on the theoretical aspects of approximation, in particular for the rational case, is deep and mature. Methods that can be used for practical purposes, on the other hand, seem very specialised, too complex, or prone to error. We hope that the methods presented in this thesis have the potential to bring some of the powerful results that have been developed in the theory to the range of application.

CHAPTER 2

Approximation by Polynomial Interpolants

About three years after *Interpolation and Approximation* appeared, I received, from a computer scientist in one of the Scottish universities, a letter that went along these lines:
" I should like to tell you how much I and my colleagues at St. Ida's are grateful to you for placing in our hands a work which is distinguished for its masterful etc. etc." (Go ahead. Praise me. I can stand it.)
"However {Ah, yes. We come now to the However. Having concluded the Gloria and the Benedictus, we come to the However, the tithe that solid thinking demands of vanity}, your presentation is flawed by your insistence on spelling Chebyshev's name as 'Tschebyscheff'. This barbaric, Teutonic, non-standard orthography will gain you no friends. I sincerely hope that when you come to prepare the second edition of your book you will alter this incorrect and irritating spelling. Yours faithfully, John Begg, Professor of Mathematics."

Philip J. Davis
The Thread: A Mathematical Yarn (1989)

We begin by considering the following classical problem of approximation by polynomial interpolants: Given $\mathbf{x} = [x_0, \ldots, x_N]^\mathsf{T}$ and $\mathbf{f} = [f_0, \ldots, f_N]^\mathsf{T}$, both in $\mathbb{C}^{N+1}$, with $x_i \neq x_j$ if $i \neq j$, determine a function $p \in \mathcal{P}(N)$ such that

$$p(x_j) = f_j, \quad \text{for } j = 0, \ldots, N. \tag{2.1}$$

We refer to $\mathbf{x}$ as the *set of nodes* or the *grid*.

The Lagrange representation of the polynomial (see Section 2.1.3) guarantees its existence, and its uniqueness follows by noting that the difference between two solutions of this problem is a polynomial in $\mathcal{P}(N)$, which vanishes at least in the $N+1$ points of the grid, and is therefore is zero.

More elaborate conditions can of course be imposed on the polynomial interpolant. A well-known example is Hermite interpolation, in which confluent points of the grid correspond to a matching of higher derivatives of the underlying function (thus the truncation of a Taylor expansion is a polynomial interpolant at the origin in the Hermite sense). Other variations include interpolation in the sense of Lidston (a special case of Hermite interpolation with two grid points), Birkhoff (the information of some of the derivatives is omitted) and Abel–Gontscharoff (each node contributes to the information of a different derivative) [34, Chap. 2]. In this thesis, however, we will not discuss any of these further.

As stated in the Introduction, our interest in polynomial interpolation is related to its capabilities for approximating functions. In this work we are mainly concerned with the situation in which the values $\mathbf{f}$ come from a certain function $f : \mathbb{C} \to \mathbb{C}$ and the polynomial interpolant is constructed to approximate f on a certain compact set (typically the interval $\mathcal{I} = [-1, 1]$, or the unit circle $\mathcal{S} = \{z \in \mathbb{C} : |z| = 1\}$), with respect to the maximum norm $\| \cdot \|_\infty$. Since this is the most frequently used norm throughout this thesis, we drop the infinity subscript and simply write $\| \cdot \| = \| \cdot \|_\infty$. The approximation procedure consists of increasing the number of grid points (together with the corresponding function samples) with the prospect of the sequence of associated polynomial interpolants converging to f. For this purpose we introduce the notion of a *triangular sequence of points* $\mathbf{X}$, which consists of an infinite array

$$\mathbf{X} = \begin{array}{cccccc} x_0^{(0)} & & & & & \\ x_0^{(1)} & x_1^{(1)} & & & & \\ x_0^{(2)} & x_1^{(2)} & x_2^{(2)} & & & \\ \vdots & \vdots & \vdots & \ddots & & \\ x_0^{(N)} & x_1^{(N)} & x_2^{(N)} & \cdots & x_N^{(N)} & \\ \vdots & \vdots & \vdots & & \vdots & \ddots \end{array}$$

the Nth row of which is a grid $\mathbf{x}$ of size N. When referring to a sequence of polynomial interpolants constructed from a triangular sequence, we use a subscript to denote the degree of the polynomial, e.g. we write $\{p_j\}$, where $p_j \in \mathcal{P}(j)$ is a polynomial interpolant on the jth row of $\mathbf{X}$. If we refer to a specific grid of N points, as frequently happens in Section 2.1, we remove the superscript (N) to simplify the notation.

Our goal for this chapter is to provide a theoretical framework for the material covered in the whole document. The study of these aspects of polynomial interpolation is not new, and comprehensive treatments can be found in the classic texts on approximation theory by Rice [138], Cheney [27], Lorenz [93], Meinardus [112], Walsh [177], Davies [34] and Powell [130], and the forthcoming book by Trefethen [164] which has been a fundamental influence on this work. Much of our work has been related to Chebfun, a software system we describe in Chapter 3, that relies on polynomial interpolation in Chebyshev points (cf. Section 2.1.2). The work presented in this thesis goes beyond polynomials and into rational functions and the material in this chapter provides a convenient framework for introducing the ideas that we explore later in Chapters 4–6: rational interpolation and best polynomial and rational approximation.

In Section 2.1 we present the basic methods for the computation of polynomial interpolants, giving emphasis to Chebyshev polynomials as a basis for $\mathcal{P}(N)$, Chebyshev points as interpolation nodes, and the barycentric Lagrange formula as the appropriate procedure for its evaluation. In Section 2.2 we turn to the study of the theoretical properties of polynomial interpolants for the approximation of functions. We will see that both an appropriate distribution of the nodes and a certain smoothness of the target function are required for a satisfactory approximation. The approximation by polynomial interpolants can be regarded as a member of the family of operators known as projectors, which also includes the truncations of series expansions. We study in Section 2.3 some of the properties of these two examples of projectors and highlight their connection. Finally, we will see in Section 2.4 how polynomial interpolants provide basic information about the region where the function is analytic.

Notation

The following is some of the notation used in this and subsequent chapters: If A is a matrix of size $(N+1)\times(N+1)$, we denote by $A_{j:k}$, $j<k$, the submatrix of size $(N+1)\times(k-j+1)$ formed with the columns j to k of A. When using the notation $A_{j:k}^{\mathsf{T}}$ ($A_{j:k}^{*}$), we refer to the transpose (resp. the conjugate) of $A_{j:k}$. A'' denotes the matrix that is the same as A except that the top-left and bottom-right entries of A are halved. A sum with the first and last terms halved is denoted by Σ''. Given a polynomial ϕ, we write $\phi(\mathbf{x}) = [\phi(x_0),\ldots,\phi(x_N)]^{\mathsf{T}}$, and in particular when s is a power we denote $\mathbf{x}^s = [x_0^s,\ldots,x_N^s]^{\mathsf{T}}$. The entrywise product of two vectors $\mathbf{v}$ and $\mathbf{w}$ (also called Hadamard product) is denoted by $\mathbf{v}\cdot\mathbf{w}$. I denotes the identity matrix, the size of which is implied by the context, and $\operatorname{diag}(\mathbf{v})$ is a diagonal matrix with entries given by the vector $\mathbf{v}$. Other notation will be introduced along the document.

2.1 Computation of polynomial interpolants

Our first topic of study is the numerical computation of polynomial interpolants. Suppose $\{\phi_j\}$, $j=0,\ldots,N$, is a basis for $\mathcal{P}(N)$ and $\boldsymbol{\alpha} = [\alpha_0,\ldots,\alpha_N]^{\mathsf{T}}$ is the vector of coefficients of the polynomial $p\in\mathcal{P}(N)$ that interpolates the values $\mathbf{f}$ on the grid $\mathbf{x}$, i.e.

$$p(x) = \sum_{j=0}^{N}\alpha_j\phi_j(x), \quad \text{and} \quad p(x_j) = f_j, \quad j=0,\ldots,N.$$

Consider the Vandermonde-type matrix $A = [\phi_0(\mathbf{x})|\cdots|\phi_N(\mathbf{x})]$. The coefficients $\boldsymbol{\alpha}$ can be obtained by solving the $(N+1)\times(N+1)$ linear system

$$A\boldsymbol{\alpha} = \mathbf{f}. \tag{2.2}$$

The numerical aspects of the solution of the polynomial interpolation problem are given by the numerical properties of this system. As usual, we measure the sensitivity of (2.2) to perturbations by the condition number of A with respect to the p-norm $\|\cdot\|_p$ [168, Lecture 12], i.e.

$$\mathrm{cond}_p(A) = \|A^{-1}\|_p \|A\|_p.$$

In this work we only mention results involving the 2-norm and the ∞-norm. Clearly, the conditioning of A depends on the chosen basis for $\mathcal{P}(N)$ and on the grid. Additional properties such as the computational cost, the numerical stability of the algorithm, and the possibility of recycling computations are also related to the choice of the basis.

In this section we are concerned about the numerical computation of interpolants but not about their effectiveness for approximation (in fact we do not even suppose that the values $\mathbf{f}$ come from a particular function). As we will see in Section 2.2, the choice of nodes and the smoothness of the underlying function are the fundamental elements which determine the approximation properties of polynomial interpolants.

2.1.1 Monomials and interpolation in the unit circle

A common choice for a polynomial basis is $\phi_j(x) = x^j$, which is referred to as the *power basis* or simply the *monomials*. In this case $A = [\mathbf{x}^0|\cdots|\mathbf{x}^N] \in \mathbb{C}^{(N+1)\times(N+1)}$ is the standard Vandermonde matrix. Monomials form a suitable basis for the representation of polynomials defined on the unit circle $\mathcal{S}$. Consider the case when the grid is the set $\mathbf{z} = [z_0, \dots, z_N]^\mathsf{T}$ of the $(N+1)$-st *roots of unity*, that is,

$$z_j = \exp(2\pi i j/(N+1)), \quad j = 0, \dots, N. \tag{2.3}$$

In this case, the matrix A transforms monomial coefficients of a polynomial interpolant into its values at roots of unity. Since monomials satisfy the discrete orthogonality property on $\mathbf{z}$ given by

$$\sum_{s=0}^{N} z_s^j \bar{z}_s^k = \sum_{s=0}^{N} \exp(2\pi i s(j-k)) = \begin{cases} N+1, & \text{if } j = k, \\ 0, & \text{if } j \neq k, \end{cases} \tag{2.4}$$

it follows that $A^*A = (N+1)I$, and A^* maps values at roots of unity of a polynomial into its (scaled) monomial coefficients. The (scaled) matrix A is therefore unitary and

$$\mathrm{cond}_2(A) = 1.$$

Moreover, it can be easily shown that [52, Example 6.4]

$$\mathrm{cond}_\infty(A) = N + 1.$$

Because of their excellent numerical conditioning, monomials allow a fast solution of the system (2.2) when using a grid of roots of unity. The matrix-vector multiplication $A^*\mathbf{f}$

computes $\tilde{\mathbf{f}} = [\tilde{f}_0, \dots, \tilde{f}_N]^\mathsf{T}$, the Discrete Fourier Transform (DFT) of $\mathbf{f}$, that is

$$\tilde{f}_k = \sum_{j=0}^{N} z_j^k f_j.$$

In particular, A^* is usually known in the literature as the $(N+1)$-point DFT matrix [174]. An early reference about the connection between the DFT and polynomial interpolation in roots of unity, as well as its numerical robustness, is [155]. The following Matlab code computes the monomial coefficients of the polynomial interpolant of $\cos(z)$ at 19 roots of unity. We present the even coefficients and compare them with the first even Taylor coefficients. We explain this agreement in Section 2.3.

```
% interp_roots.m Coeffs of polynomial interpolant of cos(x) at roots of unity

fh = @(z) cos(z);                            % target function
N = 18;                                      % N+1 = # of nodes
z = exp(2i*pi*(0:N)/(N+1)).';                % roots of unity
f = fh(z);                                   % function values
c = real(fft(f))/(N+1);                      % compute A'f
c_taylor = (-1).^(0:9)./factorial(0:2:18);  % Taylor coefficients
```

```
        N = 18           Taylor coeff
   1.000000000000000   1.000000000000000
  -0.500000000000000  -0.500000000000000
   0.041666666666667   0.041666666666667
  -0.001388888888889  -0.001388888888889
   0.000024801587302   0.000024801587302
  -0.000000275573192  -0.000000275573192
   0.000000002087676   0.000000002087676
  -0.000000000011471  -0.000000000011471
   0.000000000000048   0.000000000000048
  -0.000000000000000  -0.000000000000000
```

Roots of unity are not the only set of nodes where the monomials exhibit good behaviour. For example, when considering nodes given by a Van der Corput sequence on the circle, we have [31]

$$\mathrm{cond}_2(A) < \sqrt{2N}.$$

The convenience of working with monomials vanishes when the nodes are restricted to an interval of the real line: In this case, regardless of the point distribution, the conditioning of the Vandermonde matrix grows exponentially. More specifically, for a grid of real nodes symmetric with respect to the origin, we have that [56]

$$\mathrm{cond}_\infty(A) \geq 2^{N/2}. \tag{2.5}$$

Since it is conjectured that optimally conditioned Vandermonde matrices are constructed from symmetric grids [56], it is likely that such an exponential growth of the condition number holds for any grid of real points.

2.1.2 Chebyshev polynomials and interpolation in a real interval

For nodes on the real axis a better basis is given by the Chebyshev polynomials, defined by

$$T_N(x) = \cos(N \arccos x), \quad N = 0, 1, \ldots, \quad x \in [-1, 1]. \tag{2.6}$$

Chebyshev polynomials, named after Pafnuty Chebyshev, have been the subject of extensive research and we mention here only a handful of their properties we use in this thesis (for proofs and thorough studies of Chebyshev polynomials, see the books by Fox and Parker [50], Rivlin [140] and Mason and Handscomb [104]). Since

$$\cos\big((N+1)\theta\big) + \cos\big((N-1)\theta\big) = 2\cos(N\theta)\cos\theta, \quad \theta = \cos^{-1} x,$$

and $T_0(x) = 1$ and $T_1(x) = x$, we obtain the recurrence formula

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

from which it follows that expression (2.6) is a polynomial of degree N with leading term $2^{N-1}x^N$. The formula for Chebyshev polynomials is modified for the general interval $x \in [a, b]$, $a < b$, with the Chebyshev polynomial $T_N(s)$, where s is the linear transformation

$$s = \frac{2x - (a+b)}{b-a}. \tag{2.7}$$

From (2.6), it follows that T_N satisfies $-1 \leq T_N(x) \leq 1$ for $x \in [-1, 1]$. The product of two Chebyshev polynomials is given by

$$T_m(x)T_n(x) = \frac{1}{2}(T_{m+n}(x) + T_{|m-n|}(x)).$$

The $N+1$ *Chebyshev points of the first kind* $\mathbf{y}^{(1)} = [y_0^{(1)}, \ldots, y_N^{(1)}]^\mathsf{T}$ are the zeros of $T_{N+1}(x)$, given by

$$y_j^{(1)} = \cos\frac{(2j+1)\pi}{2N+2}. \tag{2.8}$$

Also of importance are the $N+1$ *Chebyshev points of the second kind* $\mathbf{y}^{(2)} = [y_0^{(2)}, \ldots, y_N^{(2)}]^\mathsf{T}$, which are the local extrema in $[-1, 1]$ of $T_N(x)$, given by

$$y_j^{(2)} = \cos\frac{j\pi}{N}, \quad j = 0, \ldots, N. \tag{2.9}$$

In this work the grid $\mathbf{y}^{(2)}$ is of particular importance and we will make constant reference to these points, hence we simply refer to them as *Chebyshev points* and denote them as $\mathbf{y}$. Besides the polynomials defined by (2.6), which can also be found in the literature as Chebyshev polynomials of the first kind, there are other three families with the name "Chebyshev". They all share similar properties and are related between themselves in different ways (see in particular [104]). We only refer to the *Chebyshev polynomial of the second kind* of degree N, given by

$$U_N(x) = \frac{\sin\big((N+1)\arccos x\big)}{\sin(\arccos x)}, \quad x \in [-1,1].$$

Notice that the Chebyshev points correspond to the roots of U_{N-1} together with 1 and -1, i.e. the Chebyshev points are the roots of the polynomial $(1-x^2)U_{N-1}(x)$.

Chebyshev polynomials can be continued as polynomials of a complex variable z in the following way. If z is the image under the Joukowski transform of a point w in the circle of radius $\rho > 1$, i.e.

$$z = \frac{1}{2}(w + w^{-1}),$$

(which is equivalent to define $w = z + \sqrt{z^2 - 1}$, where the square root is chosen such that $|w| = \rho > 1$), then the Chebyshev polynomials can be defined as

$$T_N(z) = \frac{1}{2}(w^N + w^{-N}), \tag{2.10}$$

and the Chebyshev polynomial of the second kind as

$$U_{N-1}(z) = \frac{w^N - w^{-N}}{w - w^{-1}}. \tag{2.11}$$

Figure 2.1 shows plots of Chebyshev polynomials in the complex plane. Notice the ellipse-shaped contours that are formed, which play a crucial role in the convergence analysis of polynomial interpolants we present in Section 2.2. In particular, the following definition will be useful: The ρ-ellipse E_ρ (also known as a *Bernstein ellipse* in honour of Sergei Bernstein) is the image of the circle $|w| = \rho$ under the Joukowski transform, i.e.

$$E_\rho = \{z = \frac{1}{2}(w + w^{-1}) \in \mathbb{C} : |w| = \rho > 1\}.$$

Thus, E_ρ is an ellipse in the complex plane with foci ± 1 and such that the sum of its semimajor axis and semiminor axis is ρ. We let $\overline{E}_\rho$ denote the closed region bounded by this ellipse.

We now consider the matrix A of system (2.2) assembled with Chebyshev polynomials, i.e. $\phi_j(x) = T_j(x)$. Estimates of the condition number of A in this case were given by Reichel and Opfer [134]. For Chebyshev points, one estimate is

$$\text{cond}_\infty(A) \leq 40N^{5+\log_2 N}$$

which, although pessimistic, shows that the growth of the condition number is slower than the bound (2.5) for monomials and real nodes. Furthermore, from numerical experiments they observed that for Chebyshev points.

$$\text{cond}_2(A) < N^{0.45\log(N)}.$$

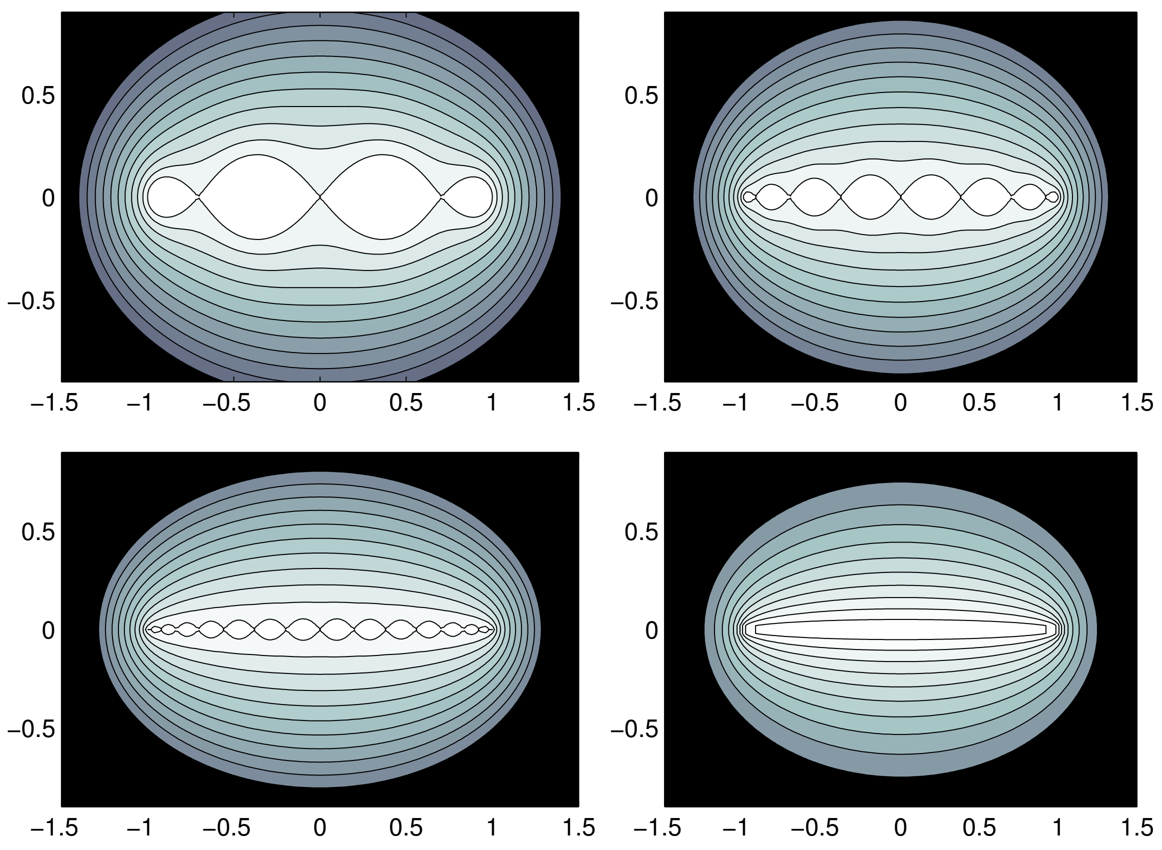

Figure 2.1: Contour plots of $|T_N(z)|^{1/N} = 1, 1.1, \ldots, 2$ (from inner to outer curves) from definition (2.10) with $N = 5$ (top-left), $N = 9$ (top-right) and $N = 17$ (bottom-left). As $N \to \infty$, $|T_N(z)|^{1/N}$ tends to $|w|$ (bottom-right, cf. Theorems 2.5 and 2.6).

The use of Chebyshev polynomials can still result in an ill-conditioned system for arbitrary sets of points [1, 38]. Figure 2.2 present the computed value of $\mathrm{cond}_\infty A$ for monomials and Chebyshev polynomials evaluated in grids of Chebyshev points and equispaced points, i.e. $\mathbf{t} = [t_0, \ldots, t_N]^\mathsf{T}$, with

$$t_j = -1 + \frac{2j}{N}, \quad j = 0, \ldots, N.$$

The efficient implementation using the FFT algorithm for polynomial interpolation in roots of unity, presented above on p. 12, can be repeated for interpolation in Chebyshev points. In this case, the appropriate tool to accelerate the computation is the Discrete Cosine Transform (DCT), which computes the matrix-vector product $A^*\mathbf{x}$ when the columns of A are Chebyshev polynomials evaluated in Chebyshev points of the first kind given in (2.8).

The following Matlab code, `interp_chebpts.m`, computes the coefficients of the polynomial interpolants of $\arccos(x)$ at Chebyshev points of the first kind. We show in the first two columns the first ten odd coefficients when $N = 50$ and $N = 5000$, the latter taking

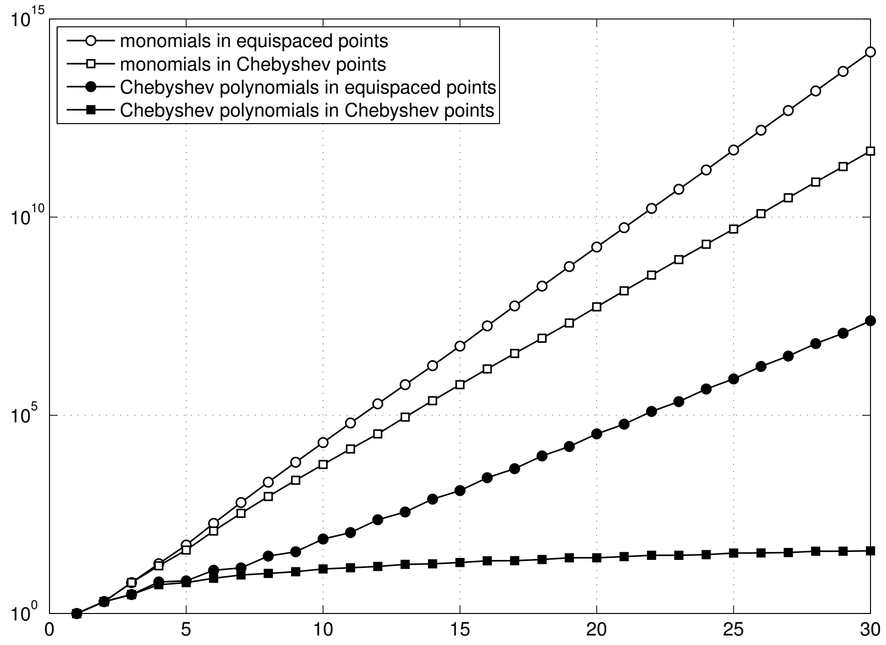

Figure 2.2: Growth of $\text{cond}_\infty(A)$ for the matrix A of system (2.2) constructed with: monomials in equispaced points (white dots), monomials in Chebyshev points (white squares), Chebyshev polynomials in equispaced points (black dots) and Chebyshev polynomials in Chebyshev points (black squares).

only a fraction of a second due to fast computation of the DCT with the FFT algorithm[1]. We also present in the third column the first ten odd Chebyshev coefficients of $\arccos(x)$ (cf. Section 2.3). As we take more interpolation points the coefficients of the interpolant converge to the Chebyshev coefficients, similarly as for the case of roots of unity and Taylor coefficients in the code `interp_roots.m` in p. 12. For this function, however, the convergence is very slow and by going from 50 to 5000 nodes we only improve the accuracy from 4 to 8 digits. We explain this phenomenon in Section 2.3.

```
% interp_chebpts.m Coefficients of poly interp of acos(x) at Chebyshev points

fh = @(x) acos(x);                                % target function
N = 50;                                           % N+1 = # of nodes
x = cos((2*(0:N)+1)*pi/(2*N+2))';                 % Cheb pts of 1st kind
f = fh(x);                                        % function values
c = dct(f)*sqrt(2/(N+1)); c(1) = c(1)/sqrt(2);    % compute A'f
c_chebyshev = [pi/2 -4/pi./(1:2:20).^2]';         % Chebyshev coefficients
```

[1]The Matlab command `dct` is only available in the Signal Processing Toolbox but it can be replaced by the following three lines written by Pedro Gonnet (where `x` and `y` are the input and output vectors respectively):

```
n = size(x,1); w = (exp(-1i*(0:n-1)*pi/(2*n))/sqrt(2*n)).'; w(1) = w(1)/sqrt(2);
if mod(n,2) == 1 ||  isreal(x), y = fft([x;x(n:-1:1,:)]); y = diag(w)*y(1:n,:);
else y = fft([x(1:2:n,:);x(n:-2:2,:)]); y = diag(2*w)*y; end, if isreal(x),y = real(y); end
```

```
        N = 50              N = 5000            Cheb coeff
  -1.273038171166869  -1.273239523799585  -1.273239544735163
  -0.141269151358525  -0.141471039590547  -0.141471060526129
  -0.050726597123964  -0.050929560853812  -0.050929581789407
  -0.025779871773988  -0.025984459569188  -0.025984480504799
  -0.015512212179501  -0.015718985789491  -0.015719006725125
  -0.010313080841158  -0.010522619929916  -0.010522640865580
  -0.007321033686225  -0.007533940867468  -0.007533961803167
  -0.005441915218702  -0.005658821485306  -0.005658842421045
  -0.004184081578395  -0.004405652229389  -0.004405673165174
  -0.003300017228865  -0.003526958412460  -0.003526979348297
```

2.1.3 Lagrange interpolation and the barycentric formula

Given a grid $\mathbf{x}$, consider now the basis $\{\ell_j(x)\}$, $j = 0, \ldots, N$, of $N+1$ polynomials in $\mathcal{P}(N)$ such that

$$\ell_j(x_k) = \begin{cases} 1, & j = k, \\ 0, & j \neq k. \end{cases}$$

Notice that this basis is not fixed in advance but depends on the data, and is not hierarchical in the sense that $\ell_j \notin \mathcal{P}(j)$. The Vandermonde-type matrix $A = [\ell_0(\mathbf{x})|\cdots|\ell_N(\mathbf{x})]$ reduces to the identity matrix and the coefficients of the polynomial interpolant specified by (2.2) are the function values to be interpolated, i.e. the polynomial interpolant is

$$p(x) = \sum_{j=0}^{N} \ell_j(x) f_j. \tag{2.12}$$

The idea of using $\{\ell_j\}$ as a basis goes back to Edward Waring in 1779, Leonhard Euler in 1783, and Joseph-Louis Lagrange in 1795 after whom the polynomials ℓ_j were named.

From their definition, it is clear that Lagrange polynomials can be written as

$$\ell_j(x) = \prod_{\substack{k=0 \\ k\neq j}}^{N} \frac{x - x_k}{x_j - x_k}. \tag{2.13}$$

A more useful representation, however, is the following: Consider the *node polynomial* on a grid $\mathbf{x}$

$$\ell(x) = (x - x_0)(x - x_1)\cdots(x - x_N), \tag{2.14}$$

and the $N+1$ *barycentric weights*

$$w_j = \frac{1}{\ell'(x_j)} = \frac{1}{\prod\limits_{k\neq j}(x_j - x_k)}, \quad j = 0, \ldots, N. \tag{2.15}$$

It is straightforward then to write (2.13) as

$$\ell_j(x) = \ell(x)\frac{w_j}{x - x_j}, \tag{2.16}$$

and the polynomial interpolant (2.12) becomes

$$p(x) = \ell(x) \sum_{j=0}^{N} \frac{w_j f_j}{x - x_j}, \tag{2.17}$$

a formula that goes back to Jacobi in 1825 [17]. With this modification, the polynomial interpolant can be evaluated and updated in $\mathcal{O}(N)$ operations, as long as the weights (2.15) are known in advance, instead of $\mathcal{O}(N^2)$ required by the formula (2.13). The numerical stability of formula (2.17) was first studied by Rack and Reimer [133] and later by Higham [78], who proved that it is backward stable (cf. [77]) independently of the grid.

Despite its robustness, formula (2.17) is affected by the growth of the barycentric weights at the rate τ^{-N}, where the value τ is known as the *logarithmic capacity of the set.* In case of an interval $[a, b]$, $\tau = (b - a)/4$, and for a circle of radius ρ, $\tau = \rho$. For example, an explicit expression of (2.15) for $(N + 1)$ Chebyshev points of the first kind on $[a, b]$ is [75]

$$w_j = (-1)^j \frac{4^N}{N(b-a)^N} \sin\frac{(2j+1)\pi}{2N+2}, \tag{2.18}$$

for $(N + 1)$ Chebyshev points (of the second kind) on $[a, b]$,

$$w_j = (-1)^j \frac{4^{N-1}}{N(b-a)^{N-1}} \delta_j, \quad \delta_j = \begin{cases} 1/2, & \text{if } j = 0 \text{ or } j = N, \\ 1, & \text{otherwise,} \end{cases} \tag{2.19}$$

and for $(N + 1)$ equispaced points on a complex circle of radius ρ,

$$w_j = \rho^N z_j. \tag{2.20}$$

Thus, if interpolating in $[-2, 2]$, the standard formula for the barycentric weights can be used directly, even for very large N. However, when working with high degrees in intervals where the logarithmic capacity is far from one (for example when $N > 500$ and $I = [-1, 1]$), the large values of w_j will cause overflow or underflow (see the top panels of Figure 2.3).

A scale-invariant alternative of formula (2.17) is obtained by dividing it by the polynomial interpolant of $f_j = 1$ for all j, and obtaining [14]

$$p(x) = \frac{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_k} f_j}{\displaystyle\sum_{j=0}^{n} \frac{w_j}{x - x_k}}. \tag{2.21}$$

We then multiply each difference in (2.15) by τ^{-1}, scaling the barycentric weights roughly to size $\mathcal{O}(1)$ and making them representable in floating point arithmetic. The weights in the numerator and denominator appear identically, except for the factors f_j, and thus common factors can be cancelled without affecting the value of $p(x)$. The downside of formula (2.21) is that it is not backward stable but only forward stable (cf. [77]) for point sets with small Lebesgue constant (cf. Section 2.3) as proved by Higham [78].

When N is even larger (for example when $N > 5000$), the computation of w_j encounters a new problem: The partial products formed along the way when the products are evaluated, for example from left to right, may overflow or underflow (see the bottom panels of Figure 2.3).

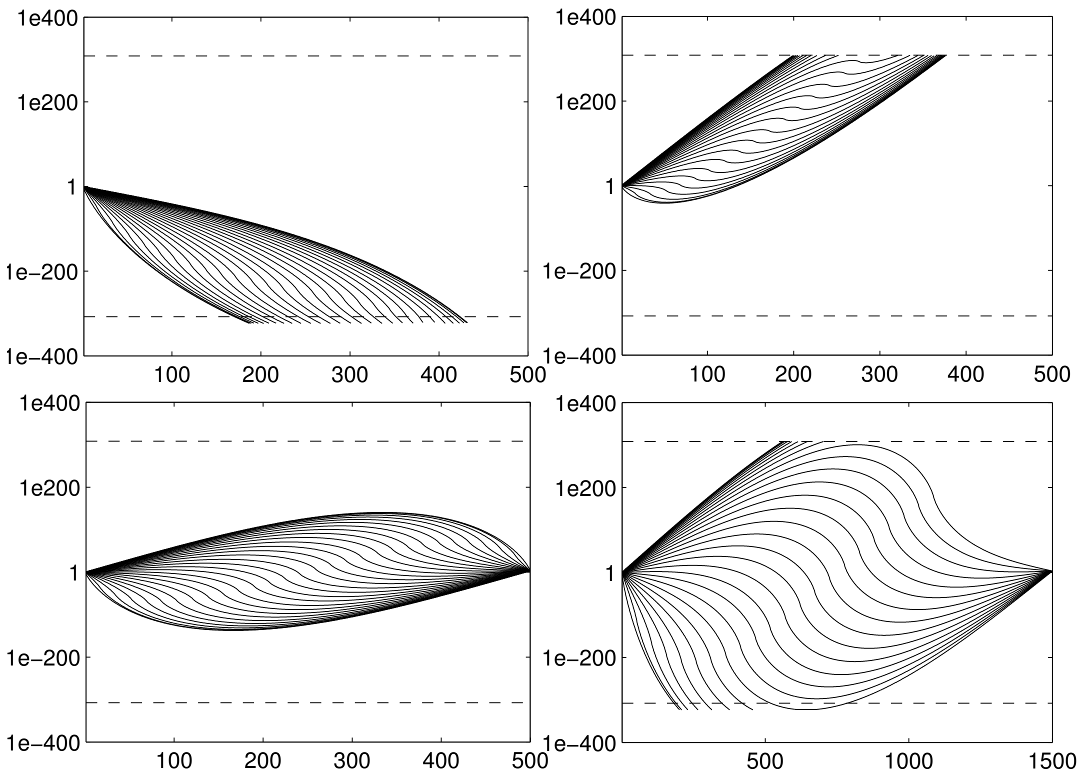

Figure 2.3: Partial products of the barycentric weights for N Chebyshev points (which are given explicitly in (2.19)) formed along the way when evaluated from left to right. Top-left: $N = 500$ in the interval $[-0.2, 0.2]$. Top-right: $N = 500$ in the interval $[-20, 20]$. Bottom-left: $N = 500$ in the interval $[-2, 2]$. Bottom-right: $N = 1500$ in the interval $[-2, 2]$. In each panel we plot the absolute value of the partial products of 30 barycentric weights in a logarithmic scale. The barycentric weights from first to last correspond to the curves, from lowest to highest. The dashed lines represent the largest and smallest normalised floating point numbers. Above and below these lines computations with numbers in double precision become `Inf` or `0`. Notice how some of the partial products stop being computed halfway through. Below the lower dashed line in the first and fourth panels, computations are carried out with denormalised numbers.

One solution to this problem would be to order the partial products to compensate, e.g. by a discrete Leja ordering [161]. However in this application we are able to use a simpler solution: rather than multiplying factors, we sum their logarithms. Following these two observations, the barycentric weights (2.15) can be replaced by

$$w_j = \frac{\displaystyle\prod_{j\neq k} \operatorname{sign}(x_j - x_k)}{\exp\Big(N \log \tau^{-1} + \displaystyle\sum_{j\neq k} \log|x_j - x_k|\Big)}, \quad j = 0, \ldots, N. \tag{2.22}$$

This is the formula used by the chebfun command `bary_weights` to compute these values in an arbitrary interval $[a, b]$ of length 4τ. For a recent survey of barycentric Lagrange interpolation, see [14].

2.1.4 Newton interpolation

The Newton basis is a common choice for representing the polynomial interpolant of values $\mathbf{f}$ on an arbitrary grid $\mathbf{x}$. In this case the polynomial interpolant is given by

$$\begin{split} p(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots + \\ f[x_0, x_1, \ldots, x_N](x - x_0)(x - x_1)\cdots(x - x_{N-1}), \end{split} \tag{2.23}$$

where the coefficients are given by the divided differences

$$f[x_j, x_{j+1}, \ldots, x_{k-1}, x_k] = \frac{f[x_{j+1}, \ldots, x_k] - f[x_j, \ldots, x_{k-1}]}{x_k - x_j},$$

with initial condition $f[x_j] = f_j$. For a discussion on Newton interpolation and some of its advantages, see [34]. In this work we will not discuss Newton interpolation further, but will use the following property in Chapters 4 and 6. Comparing the coefficients of the terms of largest degree in (2.17) and (2.23), it follows that the coefficient of degree N has the representation

$$f[x_0, x_1, \ldots, x_N] = \sum_{i=0}^{N} w_i f_i,$$

where w_i are the barycentric weights (2.15). Hence, if p is a polynomial of degree strictly less than N, then

$$\sum_{i=0}^{N} w_i p(x_i) = 0. \tag{2.24}$$

2.2 Convergence and divergence of polynomial interpolants

In this section we study the convergence properties of polynomial interpolants. Two characteristics determine the good behaviour of these approximants: the distribution of the points (of which a precise definition will be given later) and the smoothness of the underlying function. These observations have been made since the beginning of the study of approximation theory in the late 19th century and we open with a discussion on the early attempts to understand this theory, which undoubtedly marked the research focus of this subject.

2.2.1 Illusory interpolants

During the 1880s Karl Weierstrass and Charles Méray established two fundamental results in connection with the approximation properties of polynomials. The first one is the celebrated Weierstrass Approximation Theorem [179] that shows that any continuous function can be arbitrarily approximated by polynomials (see [127] for a recent discussion of this theorem):

Theorem 2.1 (Weierstrass Approximation Theorem)**.** *Let f be a continuous function in $[a, b]$. Then there exists a sequence of polynomials that converges uniformly to f on $[a, b]$.*

The conclusion of this theorem was perhaps surprising at the time, in the light of examples, constructed previously by Weierstrass himself, of quite "irregular" functions that are nowhere differentiable but still continuous [127]. The profound impact of Weierstrass theorem in Analysis can be discerned by the many different proofs that were presented afterwards. In particular, a proof given later by Bernstein [7, 90] consisted of the explicit construction of a converging sequence of polynomials by means of the *Bernstein polynomials*, defined as

$$B_k = \frac{1}{2^N}\binom{N}{k}(1-x)^{N-k}(1+x)^k, \quad k = 0, \ldots, N, \quad x \in [-1, 1].$$

Given a continuous function f, it can be shown that the sequence of polynomials

$$p(x) = \sum_{k=0}^{N} f(t_k) B_n(x), \tag{2.25}$$

where f is evaluated in a equispaced grid $\mathbf{t}$, uniformly converges to f as $N \to \infty$ [127][2]. Recursion formulas can be used to construct Bernstein polynomials of high degree, and in [46] it is shown that their computation is well-conditioned. However, the rate of convergence is merely *algebraic*, i.e. the rate of convergence is asymptotically of order $\mathcal{O}(N^{-K})$, for a positive constant K, and might be slower than desired[3] (see Figure 2.4).

At the same time as Weierstrass's result appeared, Carl Runge [142] presented a similar result for functions defined in the complex plane (see also [141, Thm. 13.7]).

Theorem 2.2 (Runge)**.** *Let $\mathcal{K}$ be a compact set in $\mathbb{C}$ such that $\mathbb{C} \setminus \mathcal{K}$ is connected. Let f be an analytic function on a neighbourhood of $\mathcal{K}$. Then there exists a sequence of polynomials that converges uniformly to f on $\mathcal{K}$.*

The requirement of connectedness on the complement of $\mathcal{K}$ is essential. Consider for example the function $1/z$ defined on the unit circle $\mathcal{S}$, on which it is analytic. Nevertheless,

[2]Another sequence of converging polynomials was given before by Borel. The polynomial basis was constructed from polynomial approximations to "hat" functions defined at each node [127].

[3]Nevertheless, Bernstein polynomials are widely used in practical applications, for example in the context of geometric design, for the formulation of what are called Bézier curves and surfaces (see [46] and the references therein).

it is not the limit of any sequence of polynomials because convergence on $\mathcal{S}$ would imply convergence to an analytic function on the unit disk $\tilde{\mathcal{S}} = \{z \in \mathbb{C} : |z| \leq 1\}$ due to the maximum principle. An analogous theorem for rational functions is mentioned in Chapter 4.

Notice that Runge's Theorem is not a true generalisation of Weierstrass theorem, since it requires analyticity not only in the interior but throughout the whole compact set $\mathcal{K}$. Thus, when $\mathcal{K}$ is a real interval, Runge's theorem requires the function to be analytic on it while Weierstrass theorem requires only its continuity. The relaxation of the conditions on the boundary of $\mathcal{K}$ was only given in 1951 by Mergelyan [116] (see also [141, Thm. 20.5]).

Theorem 2.3 (Mergelyan)**.** *Let $\mathcal{K}$ be a compact set in $\mathbb{C}$, such that $\mathbb{C} \setminus \mathcal{K}$ is connected. Let f be an analytic function in the interior of $\mathcal{K}$ and continuous on its boundary. Then there exists a sequence of polynomials that converges uniformly to f on $\mathcal{K}$.*

Figure 2.4 shows four sequences of polynomials approximating the function $(1+25x^2)^{-1}$ on $[-1, 1]$: Bernstein polynomials, best polynomial approximants, polynomial interpolants in Chebyshev points, and polynomial interpolants in equispaced points. The plots of the polynomial interpolants in the bottom panels show discrepant results: fast convergence for Chebyshev points and divergence for equispaced points.

The second crucial observation made at the end of the 19th century was that some choices of interpolants failed to converge. Méray studied and presented, in 1884 [114] and later in 1896 [115], various examples of these "illusory interpolants" that failed to converge to a desired function in various grids. In 1901, Runge [143] further explained this phenomenon, which now bears his name. In particular, he used the function $(1 + x^2)^{-1}$, $x \in [-5, 5]$, and showed that wide oscillations appear when interpolating on a grid of equispaced points despite achieving convergence in an interval around the origin. Bernstein [9] gave a more extreme example in 1918, when he showed that polynomial interpolants of $|x|$ $x \in [-1, 1]$ in equispaced points diverge at every point except $-1, 0$ and 1 (see also [63]).

The question of a grid that could satisfy convergence as predicted by Weierstrass, that is, uniform convergence for continuous functions, was still open until 1914, when Faber [45] showed the existence of a continuous function such that any sequence of polynomial interpolants fails to converge uniformly. Further evidence that the polynomials whose existence was guaranteed by Weierstrass theorem could not be merely interpolants kept appearing during the next fifty years, for example, in the work of Grünwald [69], Marcinkiewicz [101], and Erdös [44].

Despite this apparent failure, polynomial interpolants actually perform spectacularly well for smooth functions. In the next section we present results that highlight their convergence, arguing that they can be a sound choice to represent functions, as demonstrated by Chebfun.

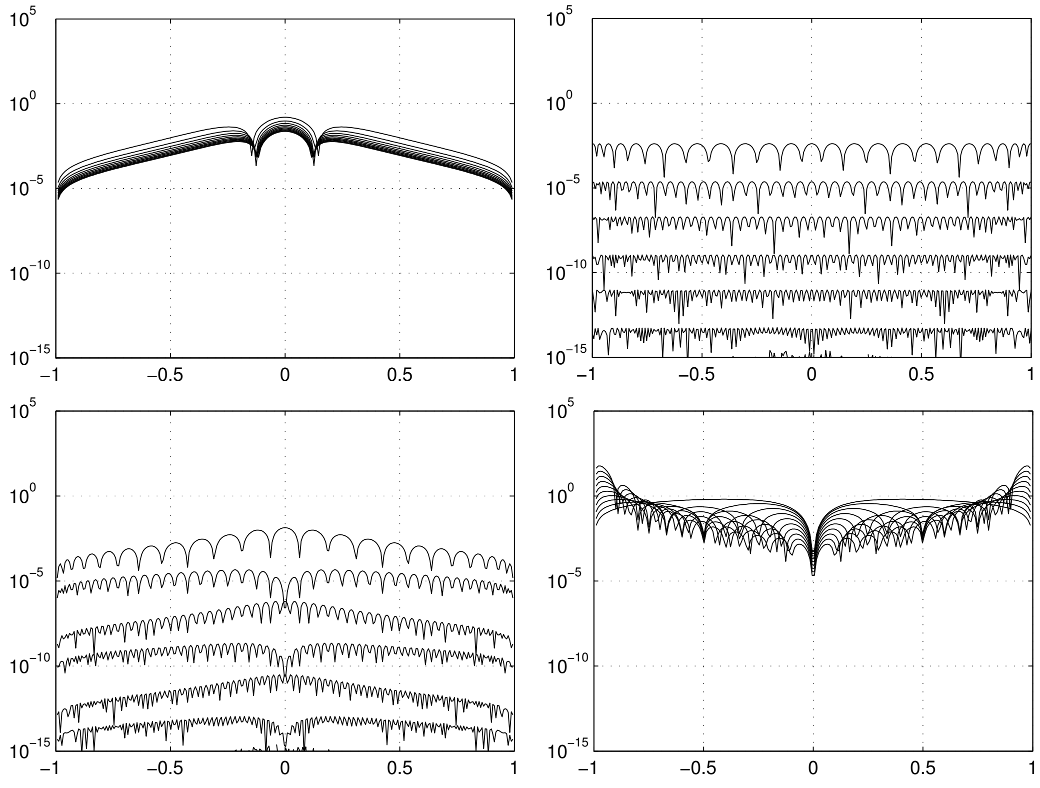

Figure 2.4: Errors of polynomial approximants to the function $(1+25x^2)^{-1}$, $x \in [-1,1]$. Top-left: Bernstein polynomials (2.25) with $N = 100, 200, \ldots, 1000$. Top-right: best polynomial approximations computed with the barycentric-Remez algorithm with $N = 25, 50, \ldots, 175$ (cf. Chapter 6). Bottom-left: polynomial interpolants in Chebyshev points with $N = 25, 50, \ldots, 175$. Bottom-right: polynomial interpolants in equispaced points with $N = 2, 4, \ldots, 20$ (notice the error growth near the endpoints).

2.2.2 Convergence for analytic functions

Most textbooks on numerical analysis give the following expression for the error of polynomial interpolation, given in terms of the derivative of a target function $f \in \mathcal{C}^{N+1}[-1,1]$:

$$f(x) - p(x) = \frac{1}{(N+1)!}\ell(x) f^{(N+1)}(\xi), \quad \xi \in [-1,1].$$

Although it is a simple formula to obtain, its applicability is rather limited since it requires an estimate of the derivative. The Hermite contour integral representation of the polynomial interpolation error [76] is a much more useful expression:

Theorem 2.4. *Let f be analytic in a simply connected region Ω and Γ be a positively oriented contour that lies in Ω and encloses the points $\mathbf{x} = [x_0, \ldots, x_N]^\mathsf{T} \in \mathbb{C}^{N+1}$, with $x_i \neq x_j$ if $i \neq j$, and the point x. Then the error of the polynomial interpolant in $\mathbf{x}$ to f is given by*

$$f(x) - p(x) = \frac{\ell(x)}{2\pi i} \int_\Gamma \frac{f(t)}{\ell(t)(t-x)} dt, \tag{2.26}$$

where $\ell(x)$ is given by (2.14).

Proof. The integrand in (2.26) has simple poles at $x, x_0, \ldots, x_N$. By Cauchy's residue theorem we have

$$\begin{aligned}\frac{1}{2\pi i}\int_\Gamma \frac{f(t)}{\ell(t)(t-x)}dt &= \operatorname{res}\Big[\frac{f(t)}{\ell(t)(t-x)};x\Big] + \sum_{j=0}^{N}\operatorname{res}\Big[\frac{f(t)}{\ell(t)(t-x)};x_j\Big]\\ &= \lim_{t\to x}\frac{f(t)}{\ell(t)} + \sum_{j=0}^{N}\lim_{t\to x_j}\frac{(t-x_j)f(t)}{\ell(t)(t-x)}\\ &= \frac{f(x)}{\ell(x)} - \sum_{j=0}^{N}\frac{w_j f(x_j)}{x-x_j},\end{aligned}$$

where w_j are the barycentric weights (2.15). Multiplying by $\ell(x)$, and from (2.17), the theorem follows. □

The Hermite integral is a fundamental tool in the convergence analysis of interpolants and we highlight its importance by presenting the proof of some theorems that use it or are closely related to it. In particular, we show how it can be employed to prove uniform convergence for analytic functions (Theorems 2.5 and 2.6) and piecewise analytic functions (Theorem 2.8), establish the number of points per wavelength required to resolve a function (Theorem 2.9), explain Runge's phenomenon (see p. 27), prove the direct result of analytic continuation for polynomials (Theorem 2.10) and the de Montessus de Ballore Theorem for rational interpolants (Theorem 5.2). We are ready to give the first proof of polynomial interpolation convergence for analytic functions.

Theorem 2.5. *Let f be an analytic function in $\overline{E}_\rho$, the closed region bounded by a Bernstein ellipse with parameter $\rho > 1$. Let $\{p_j\}$ be the sequence of polynomial interpolants associated with a triangular sequence of Chebyshev points. Then $\{p_j\}$ converges to f uniformly, and more precisely*

$$\|f - p_N\| = \mathcal{O}(\rho^{-N}). \tag{2.27}$$

Proof. Consider as the contour of the integral in (2.26) the ρ-ellipse E_ρ. From Theorem 2.4, the estimation theorem of integrals in the complex plane, the maximum modulus theorem, and $|t-x|>0$, it follows that

$$|f(x)-p(x)| \le K \max_{t\in E_\rho}\Big|\frac{\ell(x)}{\ell(t)}\Big|, \quad x\in[-1,1],\ t\in E_\rho, \tag{2.28}$$

where K is a positive constant. From the definition of $\ell(x)$ (2.14) and the relation between Chebyshev points and U_{N-1} (see p. 14), we have that $\ell(x) = (1-x^2)U_{N-1}(x)$, hence $|\ell(x)|\le 1$, $x\in[-1,1]$, and from (2.11), we obtain an expression for $\ell(z)$ when $z\in\mathbb{C}\backslash[-1,1]$:

$$\ell(z) = \Big(1-\frac{1}{4}(w+w^{-1})^2\Big)\Big(\frac{w^N-w^{-N}}{w-w^{-1}}\Big) = \frac{1}{4}(w-w^{-1})(w^{-N}-w^N),$$

where as before, $w = z + \sqrt{z^2 - 1}$ and the square root is chosen such that $|w| > 1$. Then, it is clear that

$$\limsup_{N\to\infty} |\ell(z)|^{1/N} = \limsup_{N\to\infty} |\text{Im } w|^{1/N} |\text{Im } w^N|^{1/N} = |w|,$$

and in particular

$$\limsup_{N\to\infty} |\ell(t)|^{1/N} = |w| = \rho, \quad t \in E_\rho, \tag{2.29}$$

from which the theorem follows. □

The rate at which polynomial interpolants in Chebyshev points converge in Theorem 2.5, i.e. asymptotically of order $\mathcal{O}(e^{-KN})$, where N is the number of grid points and K is some positive constant, is usually referred to in the literature as *geometric convergence.*

The fundamental aspect of the previous proof of Theorem 2.5 is that $|\ell(x)|$ remains approximately constant on the Bernstein ellipses, a property shared by other sets of nodes as well. To generalise the previous convergence result, we need to introduce some notions of potential theory (we follow the approach taken by Krylov in [86]). Consider a distribution function $\mu(x)$ which in this context is analogous to the cumulative distribution function in probability theory[4], i.e. $\mu(x)$ is a monotone nondecreasing right-continuous function, such that $\lim_{x\to-\infty} \mu(x) = 0$ and $\lim_{x\to\infty} \mu(x) = 1$. For the grid in the Nth row of a triangular sequence we assign a mass of $1/(N + 1)$ to each point, similarly as the probability mass function of a discrete random variable, and we define $\mu_N(x)$ as the total mass in $(-\infty, x]$.

A limiting distribution of a triangular sequence of points is a distribution function $\mu(x)$ such that $\mu_N(x) \to \mu(x)$ at each point of continuity of $\mu(x)$. For example, for Chebyshev points we have the *Chebyshev distribution*

$$\mu(x) = \frac{1}{\pi} \int_{-1}^{x} (1 - t^2) dt, \tag{2.30}$$

and for equispaced points we have the limiting distribution

$$\mu(x) = \frac{1}{2}(x + 1). \tag{2.31}$$

The last concept we require is that of *logarithmic potential* of a distribution $\mu(x)$, defined as

$$u(z) = \int_{-1}^{1} \log \frac{1}{|z - t|} d\mu(t).$$

In particular, for a grid in the Nth row of a triangular sequence with a distribution $\mu_N(x)$ we have

$$u_N(t) = \int_{-1}^{1} \log \frac{1}{|z - t|} d\mu_N(t) = \frac{1}{N} \sum_{j=0}^{N} \log \frac{1}{|t - x_j|}. \tag{2.32}$$

The following is a generalisation of Theorem 2.5:

[4]Notice that distributions in compact sets in the complex plane have a natural definition as positive Borel measures. In this case, convergence to a limiting distribution is in the sense of weak-* convergence. This is a useful generalisation that allows a deeper study of polynomial interpolants [91].

Theorem 2.6. *Let f be an analytic function in $\overline{E}_\rho$, the closed region bounded by a Bernstein ellipse with parameter $\rho > 1$. Let $\{p_j\}$ be the sequence of polynomial interpolants associated with a triangular sequence with limiting Chebyshev distribution* (2.30). *Then $\{p_j\}$ converges to f uniformly, and more precisely*

$$\|f - p_N\| = \mathcal{O}(\rho^{-N}).$$

Proof. We start from the error bound (2.28), which is obtained from the Hermite integral representation (2.26) using as integration contour the ρ-ellipse E_ρ. Similarly as in Theorem 2.5, we need to obtain a bound for $|\ell(x)|/|\ell(t)|$, $x \in [-1, 1]$ and $t \in E_\rho$. Expressing $\ell(x)$ as the exponential of the sum of logarithms, and from (2.32), we obtain

$$\begin{aligned}\left|\frac{\ell(x)}{\ell(t)}\right| &= \exp\Big(\sum_{j=0}^{N} \log\frac{1}{|t-x_j|} - \sum_{j=0}^{N} \log\frac{1}{|x-x_j|}\Big)\\ &= \exp\big(N(u_N(t) - u_N(x))\big).\end{aligned}$$

Since it is assumed that $\mu_N(x) \to \mu(x)$ as $N \to \infty$, it follows from Helly's theorem on the passage to the limit for Stieltjes integrals [184, p. 121] that

$$\int_{-1}^{1} \log\frac{1}{|z-t|} d\mu_N(t) \to \int_{-1}^{1} \log\frac{1}{|z-t|} d\mu(t), \quad \text{as } N \to \infty,$$

that is, $u_N(x) \to u(x)$. It follows that

$$\limsup_{N\to\infty} \left|\frac{\ell(x)}{\ell(t)}\right|^{1/N} = \exp(u(t) - u(x)). \tag{2.33}$$

The logarithmic potential of the Chebyshev distribution (2.30) is given by [86]

$$u(z) = \frac{1}{\pi}\int_{-1}^{1} \log\frac{1}{|z-t|}\frac{dt}{\sqrt{1-t^2}} = \log\frac{2}{|z+\sqrt{z^2-1}|}, \tag{2.34}$$

hence

$$u(t) = \log\frac{2}{|\rho|} \quad \text{for } t \in E_\rho, \quad \text{and} \quad u(x) = \log 2 \quad \text{for } x \in [-1, 1].$$

From (2.33) the theorem is established. □

A limiting Chebyshev distribution is not only a sufficient condition for convergence but also necessary. The following convergence result requires a more technical proof and we refer the reader to [86]:

Theorem 2.7. *Let $\mathbf{X}$ be a triangular sequence of points whose associated sequence of polynomial interpolants converges uniformly for every analytic function f on $[-1, 1]$. Then the limiting distribution function of $\mathbf{X}$ is the Chebyshev distribution* (2.30).

Theorem 2.6 provides the elements to analyse the failure of interpolants to converge on equispaced points. For any triangular sequence of points with limiting distribution $u(z)$, the crucial bound of $|\ell(x)|/|\ell(t)|$ on the error (2.28) is given by the exponentiated difference of the logarithmic potentials (2.33). The limiting distribution of the sequence of equidistant grids is (2.31) and it is easy to compute its logarithmic potential:

$$u(z) = 1 + \frac{1}{2}\mathrm{Re}[(z-1)\log(z-1) - (z+1)\log(z+1)]. \tag{2.35}$$

In Figure 2.5 we show the lens-shaped ellipses corresponding to the equipotential curves $|u(z)| = K$, where K is a positive constant. Note that that potential is not constant throughout $[-1,1]$ but is a concave function with $u(0) = 1$ and $u(\pm 1) = 1 - \log 2$. Suppose a function is analytic in the closure of the region bounded by the equipotential $|u(z)| = 1 - \log 2$, (which also crosses the imaginary axis on $\pm i0.525524914575052\ldots$). Then, a curve such that $|u(z)| < 1 - \log 2$ can be used as the contour Γ of Hermite's integral (2.26) and the proof of Theorem 2.6 remains almost unchanged: Since $u(t) < 1 - \log(2)$ and $u(t) < u(x)$, $t \in \Gamma$ and $x \in [-1,1]$, the right-hand side of (2.33) is less than 1 and uniform convergence follows.

If the function is not analytic in $\overline{E}_\rho$, the nearest singularity to $[-1,1]$ prevents convergence outside the equipotential curve that crosses it. For example, suppose a simple pole $\tilde{z}$ is located on the equipotential $|u(z)| = K_2 > 1 - \log 2$ and consider the contour $\Gamma = \Gamma_1 \cup \Gamma_2$, where Γ_1 is the equipotential $|u(z)| = K_1 < 1 - \log 2$ and Γ_2 is a small contour winding around $\tilde{z}$. Then the Hermite integral for the error gives

$$f(x) - p(x) = \frac{\ell(x)}{2\pi i}\left[\int_{\Gamma_1} \frac{f(t)}{\ell(t)(t-x)}dt + \int_{\Gamma_2} \frac{f(t)}{\ell(t)(t-x)}dt\right].$$

Following the previous analysis, the first integral on the right goes to zero and the second one is obtained by its residue. Hence we have that

$$f(x) - p(x) = \epsilon_N(x) + \frac{1}{2\pi i}\frac{\ell(x)}{\ell(\tilde{z})}\frac{g(\tilde{z})}{\tilde{z} - x},$$

where $\|\epsilon_N(x)\| \to 0$ as $N \to \infty$, and $g(z) = (z - \tilde{z})f(z)$. Hence, for $x \in [-1,1]$ such that $K_1 < u(x) < K_2$, where u is defined by (2.35), we have that $u(\tilde{z}) - u(x) > 0$ and

$$\limsup_{N\to\infty}\left|\frac{\ell(x)}{\ell(t)}\right|^{1/N} = \exp(u(t) - u(x)) > 1, \quad t \in \Gamma,$$

and the error grows exponentially. For the function $(1 + 25x^2)^{-1}$, Runge's function scaled to the interval $[-1,1]$, the singularity is located on the equipotential $|u(z)| = K = |u(i/5)|$ that crosses the real axis on $\pm x_c$, with $x_c = 0.726676860477669\ldots$. Thus, polynomial interpolation in a equispaced grid diverges on $[-1, -x_c] \cup [x_c, 1]$, and a similar argument involving the Hermite integral shows that it converges on $(-x_c, x_c)$.

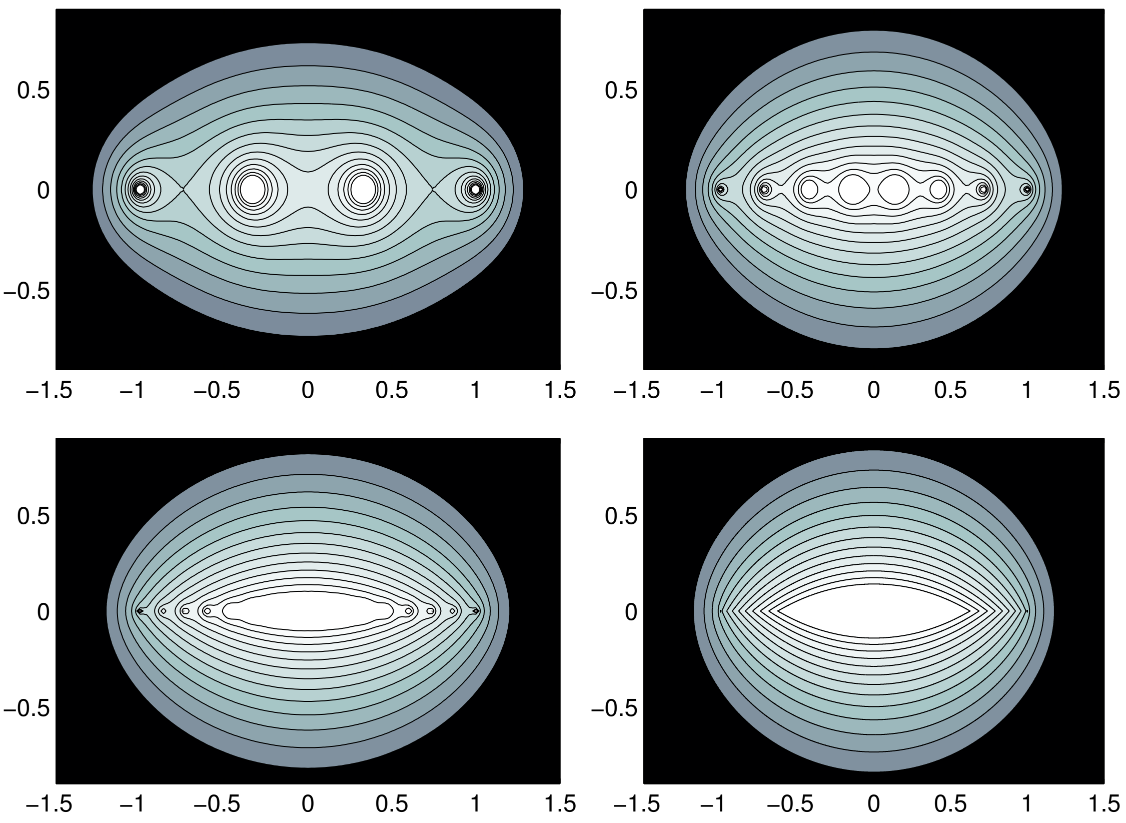

Figure 2.5: Contour plots in the complex plane of the node polynomial for equispaced points $|\ell(z)|^{1/N} = 1, 1.1, \ldots, 2.1, 2.2$ (from inner to outer curves) with $N = 4$ (top-left), $N = 8$ (top-right), and $N = 16$ (bottom-left). As $N \to \infty$, $|\ell(z)|^{1/N}$ tends to the exponential of the logarithmic potential $u(z) = 1 + \frac{1}{2}\mathrm{Re}[(z-1)\log(z-1) - (z+1)\log(z+1)]$ (bottom-right).

2.2.3 Convergence for piecewise analytic functions

Polynomial interpolants for non-analytic functions can also converge uniformly, albeit at slower rates. Consider for example a $\mathcal{C}^k$ piecewise analytic function on $[-1,1]$ with a partition $-1 = a_s < a_{s-1} < \ldots < a_0 = 1$, defined as a $\mathcal{C}^k$ function such that $f(x) = \phi_j(x)$, $x \in [a_{j+1}, a_j]$, $j = 0, \ldots, s-1$, where $\phi_j(x)$ is an analytic function on $[-1,1]$. The following theorem proved by Li [92] shows that convergence for piecewise analytic functions is algebraic. As before, the Hermite formula is used in the proof.

Theorem 2.8. *Let f be a $\mathcal{C}^k$ piecewise analytic function on $[-1,1]$ with a partition $-1 = a_s < a_{s-1} < \ldots < a_0 = 1$ and $\{p_j\}$ be the sequence of polynomial interpolants associated with a triangular sequence of Chebyshev points. Then $\{p_j\}$ converges uniformly, and more precisely*

$$|f(x) - p_N(x)| = \mathcal{O}\Big(\frac{|\ell(x)|}{N^{k+1}}\Big) \min\Big\{1, \frac{1}{N \min|x - a_j|}\Big\}, \quad \textit{for } x \in [-1,1],$$

where $\ell(x) = (1 - x^2)U_{N-1}(x)$.

Proof. Suppose that f is defined in $[a_{j+1}, a_j]$ by a function ϕ_j analytic on $[-1,1]$. Then it is straightforward to express f as the sum $f(x) = \sum_{j=0}^{s} f_j(x)$ with

$$f_j(x) = \begin{cases} \phi_j(x) - \phi_{j+1}(x), & x \geq a_j, \\ 0, & x < a_j, \end{cases} \tag{2.36}$$

for $j = 0, \ldots, s-1$ and $f_s = \phi_s$. It is easy to check that each f_j, $j = 0, \ldots, s-1$ is in $\mathcal{C}^k$ at $x = a_j$, which implies that f_j can be written as

$$f_j(x) = \begin{cases} (x - a_j)^{k+1} h_j(x), & x \geq a_j, \\ 0, & x < a_j, \end{cases} \tag{2.37}$$

for $j = 0, \ldots, s-1$, where each h_j is analytic on $[-1,1]$. For a function of the form (2.37), Cauchy's residue theorem gives the following representation for its error, derived in a similar way as (2.26) in Theorem 2.4:

$$f_j(x) - p(x) = \frac{\ell(x)}{2\pi i} \int_{\Gamma_j} \frac{(x-a_j)^{k+1} h_j(x)}{\ell(t)(t-x)} dt, \tag{2.38}$$

where Γ_j is a positively oriented contour that separates $[-1, a_j)$ and $(a_j, 1]$, contains $(a_j, 1]$ in its interior, and encloses the closure of a region where h_j is analytic. We choose as a contour Γ_j the unconnected right portion of a Bernstein ellipse together with a curve that joins its ends. More precisely, $\Gamma_j = E_{\rho,j} \cup \gamma_j$, where

$$E_{\rho,j} = \{z = \frac{1}{2}(w + w^{-1}) \in \mathbb{C} : |w| = \rho > 1 \text{ and } \text{Re } w \geq a_j\},$$

and $\gamma_j = \gamma_j^+ \cup \gamma_j^-$, where

$$\gamma_j^+ = \{z = \frac{1}{2}(w + w^{-1}) \in \mathbb{C} : w = t \exp(i\alpha_j) \text{ with } 1 \leq t \leq \rho\},$$

$\gamma_j^- = \{z : \overline{z} \in \gamma_j^+\}$, and $\alpha_j = \arccos a_j$ (see Figure 2.6). Thus

$$f_j(x) - p(x) = \frac{\ell(x)}{2\pi i} \left(\int_{E_{\rho,j}} + \int_{\gamma_j} \right) \frac{(x-a_j)^{k+1} h_j(x)}{\ell(t)(t-x)} dt.$$

Since the logarithmic potential of a Chebyshev distribution is constant on $E_{\rho,j}$, as it is constant throughout the whole Bernstein ellipse (cf. Theorem 2.5), it follows from (2.33) that the contribution to the error due to the first integral is $\mathcal{O}(\rho^{-N})$. The difference with respect to the analysis of convergence for analytic functions in Theorem 2.5 is the contribution due to the integral over the path γ_j that connects the endpoints of $E_{\rho,j}$, which can be shown to be [92, p. 294]

$$\mathcal{O}\left(\frac{|\ell(x)|}{N^{k+1}} \right) \min\left\{ 1, \frac{1}{N|x - a_j|} \right\}.$$

Thus, the algebraic rate of convergence due to the path γ_j dominates the geometric convergence due to the section of the Bernstein ellipse. Adding over j in (2.38) leads to the result of the theorem. $\square$

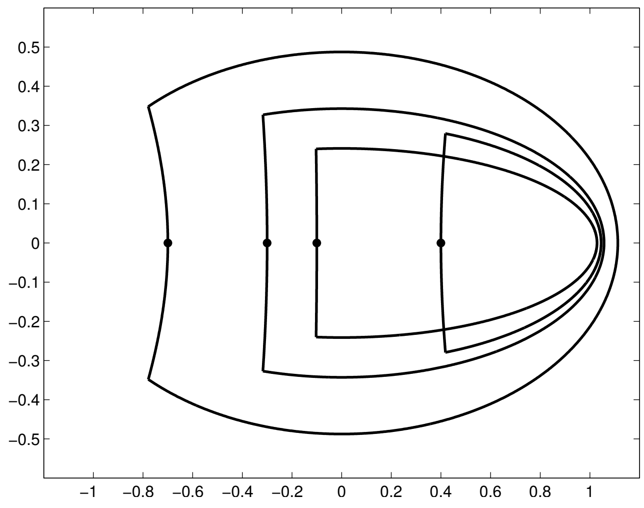

Figure 2.6: Countours Γ_j, $j = 0, \ldots, 3$ for a piecewise analytic function with a partition $a_0 = 0.4$, $a_1 = -0.1$, $a_2 = -0.3$ and $a_3 - 0.7$, where the parameters of ρ of $E_{\rho,j}$, $j = 0, \ldots, 3$ are 1.35, 1.27, 1.4 and 1.6 respectively.

The assumptions of the theorem were later relaxed by Mastroianni and Szabados [107] and Zhou and Zhu [185]. Using a different approach, Trefethen [164, Thm. 7.2] has shown that if $f, f', \ldots, f^{(\nu-1)}$ are absolutely continuous with $f^{(\nu)}$ of bounded variation V, then the polynomial interpolants in Chebyshev points satisfy

$$\|f - p_N\| \leq \frac{4V}{\pi\nu(N-\nu)^\nu}.$$

The proof of this result, instead of using the Hermite integral formula, relies on obtaining bounds for the coefficients of the Chebyshev series of f (cf. Section 2.3).

2.2.4 The Gibbs phenomenon

We have seen that polynomial interpolants in certain grids converge for analytic functions and piecewise analytic functions. How much can we relax the smoothness condition of the target function while still attaining uniform convergence? The answer is that f should satisfy the Dini–Lipschitz condition, i.e. $\omega_f(\delta) = o(|\log\delta|^{-1}|)$, where $\omega(\delta)$ is the modulus of continuity of f,

$$\omega_f(\delta) = \sup_{|x_1 - x_2| < \delta} |f(x_1) - f(x_2)|, \quad x_1, x_2 \in [-1, 1]. \tag{2.39}$$

See [160, Thm. 14.4] for a proof.

On the other hand, the polynomial interpolant in Chebyshev points oscillates near points where f is discontinuous and as $N \to \infty$, an occurrence known as the *Gibbs phenomenon*. These oscillations form an overshoot, the relative magnitude of which does not diminish with increasing N, but converges to a fixed value. For the function $\operatorname{sign}(x)$ on $[-1, 1]$, for example, this magnitude is $1.282\ldots$ when n is odd, and $1.065\ldots$ when n is even. Clearly the Gibbs phenomenon prevents uniform convergence but it can be proved that away from the discontinuity, the interpolant converges locally at a rate of $\mathcal{O}(N^{-1})$.

2.2.5 Number of points per wavelength

A well-known result in the theory of trigonometric interpolation of periodic functions is the Nyquist–Shannon sampling theorem that says that a minimum of 2 sampling points are required to "resolve" the function. An analogous condition for polynomial interpolation in Chebyshev points was given by Weideman and Trefethen [178]:

Theorem 2.9. *Let α be a real parameter and define*

$$f_N = \exp(i\alpha N x), \quad \textit{for } x \in [-1,1].$$

Let $\{p_j\}$ be the sequence of polynomial interpolants associated to a triangular sequence of Chebyshev points. Then $\{p_j\}$ converges to f uniformly as long as $|\alpha| < 1$, i.e. there are more than π interpolation points on average per wavelength.

Proof. From the Hermite integral we have

$$|f(x) - p(x)| \leq K \max_{t\in E_\rho} |f_N(t)| \max_{t\in E_\rho} \Big|\frac{\ell(x)}{\ell(t)}\Big|, \quad x \in [-1,1],\ t \in E_\rho,$$

where E_ρ is any Chebyshev ellipse with $\rho > 1$. Notice that in this case the target function varies with N, thus we need to obtain an estimate of its maximum in terms of N. If y_ρ is the real part of the point where E_ρ crosses the imaginary axis, then it is obvious that

$$\max_{t\in E_\rho} |f_N(t)| = \exp(|\alpha| N y_\rho).$$

From (2.33) we have that

$$\limsup_{N\to\infty} \left[\max_{t\in E_\rho} |f_N(t)| \Big|\frac{\ell(x)}{\ell(t)}\Big| \right]^{1/N} = \exp(|\alpha| y_\rho + u(t) + u(x)),$$

where $u(z)$ is the logarithmic potential for the Chebyshev distribution. As we have seen before $u(t) = \log(2)/|\rho|$ for $t \in E_\rho$ and $u(x) = \log 2$ for $x \in [-1,1]$. Hence, for the polynomial interpolant in Chebyshev points to converge we require the exponent in the right-hand side of the previous equation to be negative, i.e.

$$|\alpha| < \frac{\log|\rho|}{y_\rho}.$$

The maximum value attainable by $|\alpha|$ corresponds to the minimum number of points per wavelength required for convergence. Since the maximum value of the expression on the right-hand side in the previous inequality is $(\log|\rho|)/y_\rho = 1$ as $\rho \longrightarrow 1$ (i.e. when the ellipse approaches the real line and y_ρ tends to zero), and the claim of the theorem is established. $\square$

The sufficient condition of the previous theorem changes to $|\alpha| > \pi/6$ when interpolating equidistant points, i.e. there need to be more than 6 interpolation points on average per wavelength. The proof remains the same with the expression for the logarithmic potential appropriately modified.

2.3 Lebesgue constants and truncations of series

A different angle on the material presented so far is to study polynomial interpolation as a linear operator $L_{\mathbf{x}}$ from the Banach space of continuous functions to their polynomial interpolants in a grid $\mathbf{x}$ of $N+1$ points. The norm of $L_{\mathbf{x}}$, given by

$$\Lambda_{\mathbf{x}} = \|L_{\mathbf{x}}\| = \sup_f \frac{\|L_{\mathbf{x}}(f)\|}{\|f(\mathbf{x})\|} \tag{2.40}$$

is known as the *Lebesgue constant* and it is easy to see that it is the maximum of the *Lebesgue function*

$$\lambda(x) = \sum_{j=0}^{N} |\ell_j(x)|,$$

where ℓ_j is the fundamental Lagrange polynomial of degree j defined by (2.13). The Lebesgue constant is an important quantity that measures the possible growth of the interpolation error. For example, from the Banach–Steinhaus theorem, it follows that if the Lebesgue constant is not bounded for a triangular sequence of points, then there exists a continuous function for which its polynomial interpolants diverge. The work of Erdös that we mentioned in Section 2.2.1 showed that this is the case, and in particular,

$$\Lambda(\mathbf{x}) \geq \frac{2}{\pi} \log(N+1) + c,$$

where c is a positive constant and $\mathbf{x}$ is the Nth row of an arbitrary triangular sequence. From this follows the conclusion of Faber's divergence theorem also mentioned in Section 2.2.1.

Polynomial interpolation operators belong to a broader class of approximation operators known as *projectors*. These are defined as bounded, linear, and idempotent mappings from a normed linear space into a subspace of it. In the context of this chapter, the mapping is from $A(\mathcal{K})$, the space of analytic functions in the interior of a compact set $\mathcal{K}$ and continuous on its boundary, to $\mathcal{P}(N)$. Projections are *near-best* within a relative distance C in the following sense: if $p \in \mathcal{P}(N)$ is the image of f under a projection P, then

$$\|f - p\| \leq (1 + C)\|f - p^*\|,$$

where p^* is the best polynomial approximation of f of degree N and $C = \|P\|$ (in particular, for polynomial interpolants $C = \Lambda_{\mathbf{x}}$). Thus, projectors with a small norm are practical alternatives to best approximations, which usually involve relatively complex algorithms as we will see in Chapter 6.

For the case of $L_{\mathbf{z}}$ as a projector from $A(\tilde{\mathcal{S}})$ to $\mathcal{P}(N)$, where $\mathbf{z}$ is a grid of $(N+1)$st roots of unity, Gronwall [68] gave the following bound and asymptotic estimate (see also [59]):

$$\Lambda_{\mathbf{z}} \leq \frac{1}{N+1}\sum_{j=0}^{N} \csc\frac{2k+1}{2N+2}\pi \quad\text{and}\quad \Lambda_{\mathbf{z}} \sim (2/\pi)\big(\log N + \gamma + \log(2/\pi)\big) + \epsilon_N, \tag{2.41}$$

where ϵ_N tends to zero and γ is the Euler–Mascheroni constant.

Among the family of projectors from $A(\tilde{\mathcal{S}})$ to $\mathcal{P}(N)$ there is one, however, with a smallest norm: Truncations of Taylor series, i.e. if $f(z) = \sum_{k=0}^{\infty} a_k z^k$, then the *Taylor truncation of degree* N is the polynomial

$$S_T(f,N) = S_T(f) = \sum_{k=0}^{N} a_k z^k.$$

Geddes and Mason [59] showed that Taylor truncations are minimal in the sense that $\|S_T\| \leq \|P_N\|$ for all projections $P : A(\tilde{\mathcal{S}}) \to \mathcal{P}(N)$.

The study of the projector S_T goes back to Landau in 1913 [40], who gave the following exact expression for its norm, known in the literature as Landau's constant:

$$\|S_T\| = 1 + \left(\frac{1}{2}\right)^2 + \left(\frac{1\cdot 3}{2\cdot 4}\right)^2 + \cdots + \left(\frac{1\cdot 3\cdots(2N-1)}{2\cdot 4\cdots 2N}\right)^2.$$

The asymptotic behaviour of $\|S_T\|$ is also logarithmic, as proved by Landau himself, and some recent bounds are presented in [43].

Although Taylor truncations are minimal, interpolants in roots of unity are easier to compute, and from the discussion in Section 2.1.1, their coefficients can be obtained by the FFT algorithm. Moreover, due to an aliasing phenomenon [164], the coefficients of $L_{\mathbf{z}}(f)$ converge to those of S_T as N increases. This was the basis of the method proposed by Lyness and Sande [94] to compute Taylor coefficients, later improved by Fornberg [49].

Although the Lebesgue constant has at least a logarithmic growth for any set of nodes, for the sequence of Chebyshev points they satisfy [22]

$$\Lambda_{\mathbf{y}} \leq 1 + \frac{2}{\pi}\log(N+1), \quad\text{and}\quad \Lambda_{\mathbf{y}} \sim \frac{2}{\pi}\log N, \ \text{as } N \to \infty, \tag{2.42}$$

where $\mathbf{y}$ is a grid of $(N+1)$ Chebyshev points. This implies that, for example, interpolants in Chebyshev grids are only within a factor of 5 of the best approximants of degree 10,000 that we will compute in Chapter 6. Not surprisingly, a different situation occurs for equidistant grids. The Lebesgue constant grows exponentially, within the bounds [171]

$$\frac{2^{N-2}}{N^2} < \Lambda_{\mathbf{t}} < \frac{2^{N+3}}{N},$$

where $\mathbf{t}$ is a grid of $(N+1)$ equidistant points, and Turetskii (cf. [22]) gave the following asymptotic expression

$$\Lambda_{\mathbf{t}} \sim \frac{2^{N+1}}{eN\log N},$$

later improved by Schönhage and others (cf. [22]). This result has been ignored and rediscovered on various occasions as recounted in [171]. Lebesgue constants and Lebesgue functions have been studied in depth for grids in a real interval, and a lot of details are known about them. We only mention one that seems somehow surprising: The minimum local maximum of the Lebesgue function for equispaced points is equal to Landau's constant, the norm of the Taylor truncation operator.

Another example of a projector is the *Chebyshev truncation.* Let f be a Lipschitz continuous function on $[-1,1]$ that can be expressed as the Chebyshev series

$$f(x) = \sum_{k=0}^{\infty}{}' c_k T_k(x), \quad \text{with} \quad c_k = \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx,$$

where the prime on the sum denotes that the first term is halved. A Chebyshev truncation of degree N is the polynomial[5]

$$S_C(f,N) = S_C(f) = \sum_{i=0}^{N}{}' c_k T_k(x).$$

Unlike Taylor truncations on the unit disk, Chebyshev truncations are not minimal among projectors in the interval $[-1,1]$. Nevertheless, a relation between the coefficients of $L_{\mathbf{y}}$ and those of S_C can be established, which again can be explained by aliasing (see [164, Chp. 4]). In particular, the coefficients of Chebyshev interpolants converge to those of Chebyshev truncations. This was the basis of the algorithm by Geddes to compute Chebyshev coefficients [57].

We end this section with the following remark: The theorems that we presented in Section 2.2 on convergence of polynomial interpolants have analogues for truncations, e.g. Chebyshev truncations converge geometrically for analytic functions, algebraically for smooth functions of bounded variation, and a Gibbs overshoot appears for discontinuous functions. The main technique to establish these results consists in obtaining bounds on the coefficients of the truncation.

2.4 Continuation from polynomial interpolants

A simple approach for the computation of the analytic continuation of a function f is to construct a sequence of polynomials $\{p_j\}$ that converges to f on $[-1,1]$, and obtain estimates for the error $\|f - p_N\|_{E_\rho}$ as $N \to \infty$, where $\|\cdot\|_{E_\rho}$ denotes the supremum norm on a ρ-ellipse on the complex plane. The following "direct result" generalises Theorem 2.5 by showing that polynomial interpolants in Chebyshev points also converge uniformly on ρ'-ellipses, although the base of the geometric rate at which it does so depends on the distance between $E_{\rho'}$ and the closest one containing a singularity.

[5]Notice that, although an integrable function that is not Lipschitz may not have a convergent Chebyshev series, its Chebyshev truncation can be formally constructed with these formulas.

Theorem 2.10. *Let f be an analytic function in $\overline{E}_\rho$, the closed region bounded by a Bernstein ellipse with parameter $\rho > 1$. Let $\{p_j\}$ be the sequence of polynomial interpolants associated with a triangular sequence of Chebyshev points. Then $\{p_j\}$ converges to f uniformly in the interior of that region, and more precisely,*

$$\|f - p_N\|_{E'_\rho} = \mathcal{O}\big((\rho'/\rho)^N\big), \quad \textit{for } 1 \le \rho' < \rho.$$

Proof. As in the proof of Theorem 2.6, the crucial equation is (2.33):

$$\limsup_{N\to\infty} \left|\frac{\ell(t')}{\ell(t)}\right|^{1/N} = \exp(u(t) - u(t')), \quad t \in E_\rho, t' \in E_{\rho'}.$$

From the logarithmic potential (2.34) for the Chebyshev distribution , and since $\rho' < \rho$, the right-hand side of the previous equation is less than one and the conclusion of the theorem follows. $\square$

Theorem 2.10 assumes that the function is analytic and concludes uniform convergence at a geometric rate. However, if we only measure the error of the polynomial approximants, can we deduce whether the function is analytic in the first place? And if it is, in what region? The following "inverse result" derives the analytic properties of the function from the information of the error, a situation more closely related to the analytic continuation problem (see [112, Thm. 75] or [93, Thm. 7] for a proof).

Theorem 2.11. *Let f be a continuous function in $[-1, 1]$ and $\{p_j\}$ be a sequence of polynomial such that*

$$\|f - p_N\| \le \mathcal{O}(\rho^{-N}), \quad \textit{for } N = 0, 1, \ldots$$

Then f can be analytically continued to an analytic function in the interior of the region bounded by the Bernstein ellipse with parameter ρ.

Notice that the inverse result is given not for a particular type of approximant, such as interpolants or truncations, but rather to any family of polynomials that approximates the function. Direct and inverse results are usually known as theorems of Jackson and Bernstein type, respectively. In Chapter 5 we come back to the problem of analytic continuation and location of singularities using rational functions.

2.5 Discussion

It should be clear from this condensed review that polynomial interpolation offers a lot of flexibility and robustness for the approximation of functions. To summarise, roots of unity or Chebyshev points (of the first or second kind), for example, are grids that should be used to interpolate functions defined on the unit circle or the unit interval, respectively. The barycentric formula is a stable procedure for evaluating the interpolant, and the FFT algorithm provides a fast method to go back and forth between interpolant values and coefficients. Convergence will depend on the smoothness of the underlying function, and if it is analytic, the rate is geometric. Moreover, the Hermite formula is a basic tool in the analysis of polynomial interpolation.

A particular aspect of the research in this thesis has been the experimentation with polynomials of very high degree. Routinely, we make computations with hundreds or thousands of points, and this type of exploration led us to look for efficient methods to implement them (for example, we got to formula (2.22) in this way).

For many years these ideas have been used for the development of algorithms, and with the introduction of Chebfun they are now exploited in a new way for the purpose of scientific computing. The theoretical background presented in this chapter allows us to understand many of the strengths of Chebfun, however it also highlights some of its problems. From Theorem 2.8, a function with discontinuities in its derivatives will converge slower than desired. Or even if the function is analytic, but with singularities too close to the real line, the ρ parameter of the Bernstein ellipse in Theorem 2.5 could be too close to 1, implying that a large number of interpolation points might be required to approximate the function to machine precision. Moreover, just as happens with any process that involves sampling, functions with high frequencies require a minimum number of points to be resolved accurately.

Overcoming some of these difficulties has been a constant goal in this thesis, and the following three chapters present the methods we have explored for this purpose: Piecewise interpolation and rational interpolation. For the first, we have focused on its implementation on Chebfun and thus far it has proven to be a reliable alternative to approximate functions with great efficiency. For the second, we have developed a new algorithm and studied its behaviour in the presence of rounding errors.

CHAPTER 3

Chebfuns and Piecewise-Smooth Chebfuns[1]

I needed some fresh air. I went out, bought myself some food, another bottle of whiskey. I came back, left the sandwiches in a corner, and started on the whiskey as I inserted the Basic disk and went to work. I made the usual mistakes, and the debugging took me a good half hour, but by two-thirty the program was functional and the seven hundred and twenty names of God were running down the screen.

Umberto Eco
Foucault's Pendulum (1988)

The application of approximation theory that drives our research is the development of Chebfun, a computing platform for the development of algorithms in which the basic operations are applied on functions rather than numbers. For the end-user, the functionality of Chebfun may resemble that of symbolic computing but with the speed of floating-point numerics[2].

Floating-point, one of the pillars of scientific computing, can be roughly described as a system for the representation of numbers by truncated strings of digits. Similar in spirit, Chebfun is a system that represents functions by piecewise polynomial interpolants, accurate to machine precision. These approximants can then be manipulated with computational operations defined on them. At each step of these computations, Chebfun automatically truncates the degrees of the polynomials to prevent their combinatorial growth, similar as truncated arithmetic operations for floating-point numbers.

[1]The material of this chapter refers mainly to Chebfun version 2 developed by L. N. Trefethen, R. Pachón, R. Platte and T. Driscoll, and is adapted from R. Pachón, R. Platte and L. N. Trefethen, *Piecewise smooth chebfuns*, IMA J. Numer. Anal. (in press) [124]. The project was envisioned and coordinated by Trefethen, the software was written almost entirely by Pachón and Platte between October 2006 and June 2008, the recursive subdivision algorithm was proposed by Trefethen and developed by Pachón, the edge detection algorithm was proposed and developed by Platte and the paper was written by Trefethen.

[2]The *motto* "Think symbolically, act numerically", coined by Driscoll, captures this aim of Chebfun.

More specifically, a chebfun[3] is a Matlab representation of a function of a continuous variable following Matlab syntax for vectors, with Matlab's vector operations overloaded by appropriate analogues[4]. Consider the following example:

```
>> f = chebfun( @(x) exp(cos(3*x)) , [0 6] );
>> g = chebfun( @(x) sin(exp(5-x)) , [0 6] );
>> [length(f) length(g)]
ans = 124   252
```

Chebfun determines here that the functions $f(x) = \exp\bigl(\cos(3x)\bigr)$, and $g(x) = \sin(\exp(5 - x))$, with $x \in [0, 6]$, can be accurately approximated by polynomial interpolants in 124 and 252 Chebyshev points respectively (information retrieved with the overloaded command `length`, which in standard Matlab shows the number of entries of a vector), and create chebfuns with all the information required to evaluate and manipulate them with approximately machine precision. Continuing with our example, the commands

```
>> x = 6*rand(1000,1);
>> max(abs( f(x) - exp(cos(3*x)) ))
ans = 5.107025913275720e-15
```

show that for one thousand random points between 0 and 6, the chebfun of f can be evaluated (with the overloaded round brackets) to an accuracy close to machine precision. The error is relative with respect to the values of the function on the whole interval and not only in the individual points.

The product of the polynomials representing f and g is of degree 374, but after truncation Chebfun finds that a polynomial of degree 264 is enough to represent $h(x) = f(x)g(x)$ to machine precision[5]:

```
>> h = f.*g; length(h)
ans = 265
```

Hundreds of Matlab commands have been overloaded in Chebfun in such a way that they provide arguably "natural" analogues for continuous functions[6]. Thus, with values typically

[3]Perhaps is useful to remark that we write 'Chebfun', with a capital letter, to refer to the software package, and 'chebfun', with lower case, to refer a member of that class in the object-oriented environment of Matlab.

[4]Two important guidelines for the development of Chebfun have been the parallel with floating point arithmetic and the correspondence with standard Matlab commands. Very frequently, when facing a dilemma about a specific aspect of Chebfun, we ask ourselves, what is the appropriate analogue of the problem in floating point arithmetic? or, what would Matlab do in this case?

[5]For a more dramatic example in which the degree of the product of polynomial approximations reduces from 19×4^{15} to 3400 by truncating at each step while preserving machine precision of the final approximant, see [166, p. 15].

[6]Notice for example that the overloaded command to take the product of two chebfuns is `.*` instead of `*`, in agreement with the pointwise product command in Matlab for vectors.

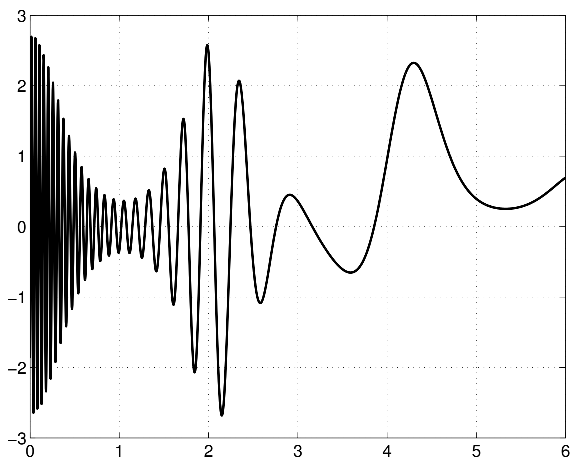

Figure 3.1: Plot of a chebfun of `length` 265.

accurate to the first 15 digits, `sum(h)` shows that the integral of h between 0 and 6 is 1.659808951734064; the command `norm(h)` computes the L_2 norm of h, 3.553935232345824; the maximum value of h, 2.718281828459047, is obtained with the command `max(h)`; and `plot(h)` produces the image in Figure 3.1.

The development of Chebfun has been a project led by Nick Trefethen at the Numerical Analysis Group of the University of Oxford. Trefethen and his former D.Phil. student Zachary Battles introduced the first ideas of the system in the version 1.0 of the software (with copyright of 2003) and the companion article "*An extension of Matlab to continuous functions and operators*", published in 2004 [6]. The original software consisted of nearly 800 lines of code for the manipulation of smooth chebfuns defined on $[-1, 1]$. Two significant contributions of Battles' thesis [5] were the powerful rootfinding technique of Chebfun and the exploration of concepts related to the linear algebra of *quasimatrices*, that is, matrices with continuous columns.

Between 2006 and 2010, Chebfun evolved further with main releases of version 2 in June 2008 and version 3 [170] in December 2009[7]. It supports piecewise smooth functions and has some capability to handle functions diverging to infinity, with arbitrary finite or infinite domains. It also includes Chebop, an extension of Chebfun for differential and integral operators[8]. In the webpage of the project, `www.maths.ox.ac.uk/chebfun` can be found a detailed guide and related publications.

In Section 3.1 we give a brief overview of some of the elements in the original Chebfun

[7]Chebfun version 2 consisted of five releases at intervals of a few months between these two dates, with version numbers 2.0296, 2.0330, 2.0399, 2.0440, and 2.0501. The number after the dot corresponds to the revision number in the revision control system used for Chebfun. The version number of the first release of Chebfun version 3 is 3.100.

[8]Chebop was conceived by Driscoll, Bornemann and Trefethen [163]. Its current development is the thesis topic of Birkisson at Oxford, who included a "nonlinear backslash" for solving ODEs.

which are still fundamental for understanding how the system operates. Some of the software aspects in the current Chebfun constructor are presented in Section 3.2. Two of the fundamental ideas of the constructor in version 2 are presented in Section 3.3, namely the recursive subdivision of functions and the edge detection algorithm. We believe that these are the aspects of Chebfun that come closest to the main theme of this thesis, that is, the approximation of functions.

3.1 Smooth chebfuns

The first stage in the construction of the chebfun of a given function is the adaptive computation of its interpolant at as many Chebyshev points as necessary to reduce the error below a certain tolerance. This was the fundamental idea introduced in the first version of Chebfun and in the current version it is implemented in the subclass fun[9].

The method used to infer the number of interpolating nodes is iterative. At the kth step, $k \geq 0$, it samples the function at a grid of $2^{k+3}+1$ Chebyshev points[10]. Then, using the FFT, it computes the Chebyshev coefficients of the polynomial interpolant as explained in Section 2.1.2. If the last *few* coefficients are *negligible*, the interpolant is accepted as being accurate enough to represent the function and the iteration stops; otherwise a new step begins. The process also stops if it reaches a maximum number of interpolation points. Figure 3.2 shows the Chebyshev coefficients for steps $k = 4, 5$, and 6 in the construction of the chebfun for the function `h` above.

From the theory presented in Chapter 2 it is reasonable to expect that for a sufficiently smooth function this process will increase the number of interpolation points up to a point in which the highest coefficients are at the tolerance level. If this is not the case, then Chebfun starts a second stage that we explain in the next section.

The method just described uses heuristics and parameters that we have been changing and tuning for the last few years. For example there is not a fixed number of coefficients taken at the tail of the expansion to verify whether they are small enough but instead this value depends on the number of interpolation points used at each step. The value used to determine whether they are negligible is not exactly machine precision but instead it depends on the length of the interval of definition and estimates of its maximum absolute value. Moreover, a special procedure is used to prune coefficients below this tolerance obtaining a representation with fewer points. The details of these aspects of the algorithm can be found in the documentation of the code.

[9]The class ‘chebfun’ in version 1 and in [6] corresponds since version 2 and in [124] to the subclass ‘fun’.

[10]The number of initial sample points is set to 9 by default. However, this parameter as well as many others can be modified by the user with the command `chebfunpref`.

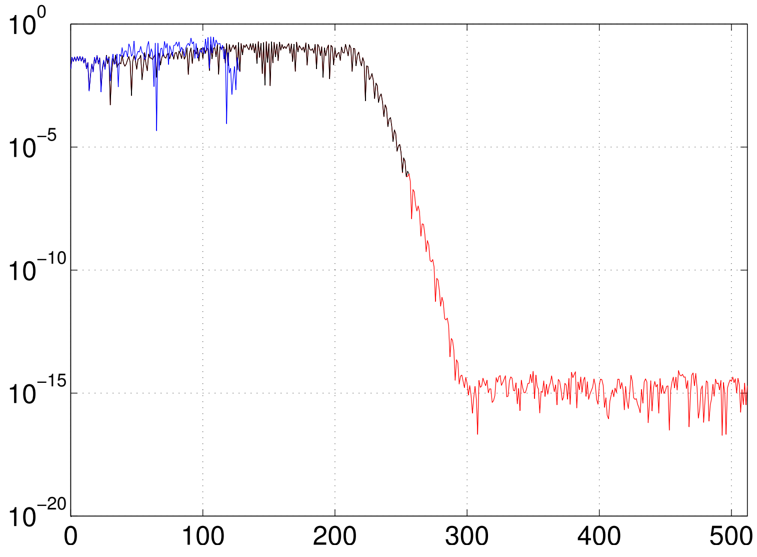

Figure 3.2: Construction of a chebfun for the function $h(x) = \exp\bigl(\cos(3x)\bigr)\sin(\exp(5-x))$. The lines show the $2^{k+3}+1$ Chebyshev coefficients of polynomial interpolants constructed at steps $k = 4$ (blue), $k = 5$ (black) and $k = 6$ (red). Notice how the first coefficients in the last two cases are almost identical (they differ by less than 10^{-7}). Since the last coefficients for $k = 6$ are negligible, Chebfun stops increasing the grid and prunes the tail.

3.2 Piecewise smooth chebfuns

Despite the good convergence properties of polynomial interpolants in Chebyshev points, many function are not easily representable in this way for practical purposes. For example, to approximate to machine precision $|x|^5, |x|^3$, and $|x|$, it would require grids of nearly $500, 4300$, and 150,000 points respectively. But even analytic functions in the complex plane may require many points if their singularities are too close to the interval of definition or if they have very high frequencies. To allow a representation of these functions in Chebfun, the system was enhanced with the capability of constructing and manipulating piecewise polynomial interpolants.

The current implementation of Chebfun consists of the definition of a new class in Matlab that contains the information required to represent a function f defined in an arbitrary interval $[a, b]$ to machine precision. The chebfun representing a function is an object that includes the fields

`funs, ends, imps`.

The field `funs` is a cell array containing n objects of the subclass fun for some positive integer n. Each fun represents a piece where f is sufficiently smooth to be effectively represented by a polynomial and each of them are constructed using the strategy described in the previous section. The field `ends` is a vector of $n + 1$ floating-point numbers in monotonically increasing order indicating subintervals,

$$a = \texttt{ends(1) < ends(2) ... < ends(n+1)} = b,$$

and the fun in `funs{i}` is defined in the interval `[ends(i),ends(i+1)]`. The field `imps` is a matrix that contains in the first row the functions values at the break points themselves, which, if the function is discontinuous, may match the fun on the left, the fun on the right or neither. Thus, in mathematical terms, we may think of a chebfun at a discontinuity as (to floating-point approximation) lower semicontinuous, upper semicontinuous or neither. Other rows of `imps` store data related to delta functions of first or higher orders, which are introduced for example if one differentiates a step function. Auxiliary information is stored in the fields `nfuns` (the value n), `scl` (a scalar equal to the largest of the absolute values of all the data defining a chebfun), `trans` (0 for a column chebfun, and 1 for a row chebfun), `jacobian` and `ID`, the last two used in Chebop.

Clearly, the crucial aspect in constructing piecewise polynomial interpolants is setting the break points, and Chebfun does this in three different ways. The simplest one is by listing the functions and their corresponding end points in the definition. For example

```
>> f = chebfun(.5,'exp(1./(1+10*x.^2))','(x-1).*cos(30*x)',[-2 -1 1 3])
f = chebfun column (3 smooth pieces)
       interval          length    endpoint values
[       -2,       -1]         1         0.5        0.5
[       -1,        1]       133         1.1        1.1
[        1,        3]        64           0       -0.9
Total length = 198   vertical scale = 2.7
```

is a chebfun with three smooth pieces in the intervals $[-2,-1]$, $[-1,1]$ and $[1,3]$, with 1, 133 and 64 Chebyshev points respectively which we plot in the left panel of Figure 3.3. The Chebfun commands are then available to compute with `f`.

The previous functionality introduces break points explicitly. However, they can also appear implicitly in overloaded commands due to their discontinuities or singularities. The following commands create a chebfun `x` in the interval $[-15,5]$ and manipulate it with the overloaded commands `real, airy, sin, abs` and `max`:

```
>> x = chebfun('x',[-15, 5]);
>> f = real(airy(x));
>> g = -.1*sin(x/4)+.03;
>> g = abs(g);
>> h = max(f,g);
```

The resulting function `h` is represented by 204 points in 16 smooth pieces with polynomials of degrees ranging from 5 to 21, defined in intervals of size between 0.095 and 3.43. In the

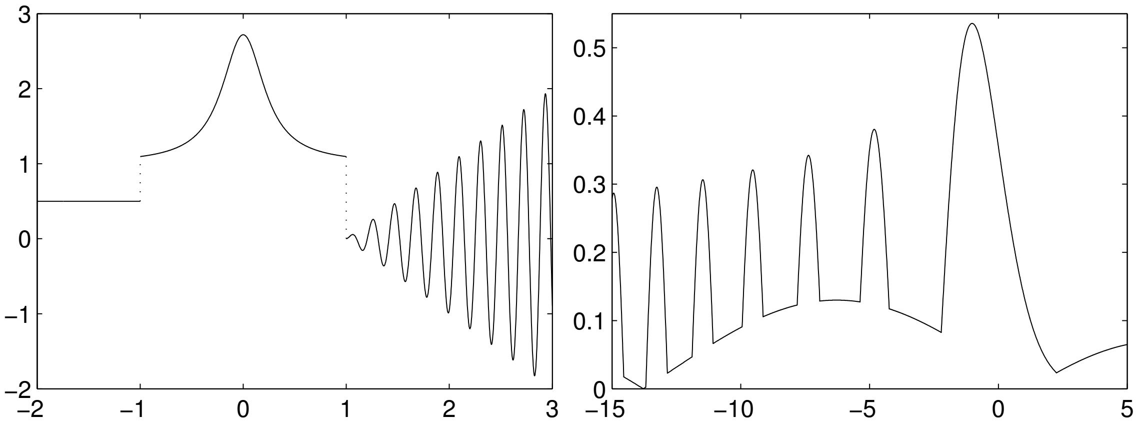

Figure 3.3: Piecewise smooth chebfuns. The break points in the chebfun at the left where introduced explicitly in the constructor and those in the chebfun at the right where introduced automatically by Chebfun commands.

right panel of Figure 3.3 we show the image of `h`. Commands that automatically introduce break points include

`abs`, `sign`, `floor`, `ceil`, `round`, `fix`, `mod`, and `rem`

when applied to one chebfun, or the commands

`min`, `max`, `conv`

when applied to two chebfuns. In the case of `abs(f)` or `sign(f)`, for example, the zeros of f on its interval of definition are first determined by the `roots` command (cf. Section 3.4). At each such zero a new break point in the chebfun is introduced, in addition to any break points that may already be there. Similarly, for `min(f,g)` or `max(f,g)`, where `f` and `g` are chebfuns defined on the same interval, the system first finds the roots of `f-g` and introduces new break points accordingly.

If the expression passed to the Chebfun constructor is that of a function with a singularity, as with `chebfun('abs(x)')`, the break point must be located and introduced automatically. This is what is known in the literature as *automatic edge detection* and in the following section we present the algorithm included in Chebfun for that purpose.

3.3 Automatic subdivision

The construction of a chebfun can be carried out in two modes which can be enabled or disabled by the user. The first one is the `splitting off` mode, in which the constructor applies the method described in Section 3.1, fitting interpolants in grids that increase their size at each step attempting to obtain a global polynomial for which its error is about machine precision. The default maximum for the number of interpolation points is $2^{16} + 1$, and if the error is above the tolerance a warning message is displayed. This mode is useful for exploratory work, for example to study approximation properties of polynomial interpolants, or for applications in differential equations, where the automatic introduction of break points may cause difficulties. The second mode of construction is `splitting on` which is useful when dealing with functions that are far from smooth. The example we gave in the Introduction in p. 2, in which we computed the best polynomial approximation to $f(x) = \exp(|x|)$, specified this mode in the syntax of the constructor.

The `splitting on` mode combines two features: recursive subdivision and edge detection. Prototypes of Chebfun before version 2 had the former but not the later. The construction of a piecewise smooth chebfun by recursive subdivision can be described in the following way. Try to construct a polynomial interpolant to the function at less than 129 Chebyshev points that represents it to the required accuracy. If this is successful, return the polynomial as the chebfun representation, otherwise split the interval in half and try again with both halves. This process is repeated until the function is represented with smooth pieces all over the domain. An additional step is introduced to avoid large numbers of spurious break points: At each step, try to merge adjacent smooth pieces.

Figure 3.4 shows the algorithm with this subdivision strategy for a function f on an interval $[a, b]$. During the construction process the chebfun F is an array of `fun`s and `naf`s, polynomials interpolating f on different subintervals, and the process finishes when these are all `fun`s. A polynomial in a subinterval is marked as a `fun` if its degree is less than 128 and accurate enough to the requirement or as a `naf`, which stands for `not-a-fun`, if not. Finally, fun$[a, b]$ stands for "`fun` or `naf` of f in the interval $[a, b]$"

The application of this algorithm on a function like $|x - 0.1|$ in $[-1, 1]$ produces two smooth pieces, each one represented by only two data points. Although this construction only takes a fraction of a second, it requires thousands of function evaluations while computing polynomial interpolants for more than a hundred different subintervals in $[-1, 1]$ and merges all of them but two. At every step, F has a `naf` in the interval $[-0.1 - \epsilon_1, 0.1 + \epsilon_2]$ and two `fun`s in the intervals $[-1, 0.1 - \epsilon_1]$ and $[0.1 + \epsilon_2, 1]$. The degree of the polynomial represented by the `naf` keeps shrinking as long as it is larger than 128, and the singularity "evaporates" after 52 iterations.

As suggested by Platte, this algorithm is greatly improved by including at each step a procedure that quickly look for discontinuities in f, f', f'' or f''' in the interval of interest. The estimates of these functions are computed using a simple finite-difference scheme rather than polynomial interpolation in Chebyshev points. If this procedure does not find a discontinuity little harm is done, since the algorithm then falls back upon the general procedure without having wasted too much effort. The algorithm for the construction of piecewise smooth chebfuns, shown in Figure 3.5, includes both the recursive subdivision and the edge detection procedures.

The **detectedge** algorithm in Figure 3.6 computes estimates of $|f'|$, $|f''|$ and $|f''''|$ from finite differences of sampled values of f of orders 1, 2, 3 and 4 on 15-point equally-spaced grids. If any of these estimates grows by more than a 50% as the grid is refined then a singularity appears to be present, and the grid is repeatedly shortened by a factor of 7 all the way down to machine precision. In case the estimate of $|f'|$ is detected to grow, then a program **findjump** uses bisection to locate the discontinuity, usually accurately to the last bit in floating-point if the jump happens in $|f'|$.

With this algorithm, Chebfun needs to evaluate the function $|x - 0.1|$ in 420 points for computing its singularity to machine precision. The algorithm can also locate multiple singularities, as happens with the following example which required the evaluation of the function at 2219 different points:

```
>> f = chebfun(@(x) sign(sin(x)),[0 10*pi],'splitting','on');
>> [f.ends' pi*(0:10)' (f.ends-pi*(0:10))']
ans =                  0                  0                  0
   3.141592653589793  3.141592653589793                  0
   6.283185307179586  6.283185307179586                  0
   9.424777960769379  9.424777960769379                  0
```

```
$F = \text{fun}[a,b]$ {Try to compute a single polynomial interpolant of the function}
while $F$ has some nafs in it do
  for every $i \geq 0$ such that $F(i)$ is a naf do
    Let $[a,b]$ be the interval associated with the $i$th naf, and $m = (a+b)/2$
    Let $L = \text{fun}[a,m]$ and $R = \text{fun}[m,b]$
    if $L \neq$ naf then
      Let $\tilde{a}$ be the left endpoint of $F(i-1)$ (if defined), and $LL = \text{fun}[\tilde{a},m]$
      if $LL \neq$ naf then
        $L = LL$, $a = \tilde{a}$
      end if
    end if
    if $R \neq$ naf then
      Let $\tilde{b}$ be the right endpoint of $F(i+1)$ (if defined), and $RR = \text{fun}[m,\tilde{b}]$
      if $RR \neq$ naf then
        $R = RR$, $b = \tilde{b}$
      end if
    end if
    Store $L$ and $R$ in $F$ as the pieces in $[a,m]$ and $[m,b]$ respectively
  end for
end while
```

Figure 3.4: Algorithm for the computation of piecewise smooth chebfuns using recursive subdivision.

```
$F = \text{fun}[a, b]$
**while** $F$ has some nafs in it **do**
  **for** every $i \geq 0$ such that $F(i)$ is a naf **do**
    Let $[a, b]$ be the interval associated with the $i$th naf,
    $m =$ **detectedge**$(f, a, b)$
    **if** $m$ is an edge at distance $\geq 10^{-14}(b - a)$ from $a$ and $b$ **then**
      Mark $m$ as a genuine edge
    **else if** $m$ is an edge at distance $< 0.01(b - a)$ from $a$ or $b$ **then**
      Set $m$ to the number at distance $0.01(b - a)$ from that endpoint
      Mark $m$ as a removable edge
    **else**
      $m = (a + b)/2$ and mark it as a removable edge
    **end if**
    $L = \text{fun}[a, m]$, $R = \text{fun}[m, b]$
    Store $L$ and $R$ in $F$ as the pieces in $[a, m]$ and $[m, b]$ respectively
  **end for**
**end while**
**for** $i$ to the number of removable edges introduced above **do**
  Merge the funs adjacent to edge$(i)$ into a single fun if possible
**end for**
```

Figure 3.5: Algorithm for the computation of piecewise smooth chebfuns using recursive subdivision and edge-detection, as appear in [124].

```
   12.566370614359172   12.566370614359172                     0
   15.707963267948966   15.707963267948966                     0
   18.849555921538759   18.849555921538759                     0
   21.991148575128552   21.991148575128552                     0
   25.132741228718345   25.132741228718345                     0
   28.274333882308138   28.274333882308138                     0
   31.415926535897931   31.415926535897931                     0
```

Notice how all the break points at $\pi, 2\pi, \ldots, 9\pi$ are correctly computed to machine precision. The algorithm splits the function into intervals where it can be represented by small degree polynomials, so not only jumps in the interval are identified. For example, the function $\sqrt{x}$ in $(0, 1]$ consists of only one 'smooth piece', being analytic throughout $(0, 1]$. Due to its singularity at $x = 0$, the function cannot be represented to machine precision by a single polynomial of reasonable degree, but the Chebfun constructor in `splitting on` mode represents it as a chebfun with seven pieces defined on exponentially graded subintervals:

```
>> f = chebfun(@(x) sqrt(x),[0 1],'splitting','on');
>> f.ends'
ans =                0
   0.000000000100000
```

```
edge = NaN
Compute estimates of $|f'|$, $|f''|$ and $|f''''|$ on a 50-point equispaced grid on the initial interval $[a, c]$
Set $b$ = gridpoint associated with the maximum estimate of $|f''''|$
$d = 4$
while $|c - a| >$ machine precision do
   Set $a$ and $c$ to the gridpoints left and right of $b$ respectively
   Compute new estimates of $|f'|, \dots, |f^{(d)}|$ on a 15-point equispaced grid on $[a, c]$
   if none of the new estimates increased by a factor of 1.5 then
      return {no edge was detected}
   end if
   Set $d$ = order of lowest derivative that increased
   Set $b$ = gridpoint associated with maximum value of $|f^{(d)}|$
   if $d = 1$ then
      $b$ = findjump$(f, a, c)$
      return
   end if
end while
edge = $b$

return edge
```

Figure 3.6: Algorithm for the detection of edges for a function f in an interval $[a, c]$. The algorithm, as published in [124], was proposed and implemented by Platte.

```
   0.000000010000000
   0.000001000000000
   0.000100000000000
   0.010000000000000
   0.505000000000000
   1.000000000000000
>> length(f)
ans = 586
>> sum(f)
ans = 0.666666666666667
```

Each subinterval is 100 times smaller than the last, and the lengths of the corresponding funs are between 19 and 120. The last line in the example shows that although the underlying representation is complicated, having 586 function values, the integral is computed to machine precision. In Chebfun version 3 the representation of $\sqrt{x}$ is simplified, as it includes the possibility of working with functions with algebraic singularities, even if they are not poles (see *Chebfun Guide 9: Infinite intervals, infinite function values, and singularities* on the webpage of the project for more on this feature).

3.4 Other commands

One of the most useful operations in Chebfun is its efficient root finder introduced by Battles since version 1 [6]. Finding the roots of a smooth chebfun is equivalent to locating the roots of a polynomial in its Chebyshev form, which in turn can be done by computing the eigenvalues of a "colleague" matrix, the elements of which are Chebyshev coefficients. This method is well-conditioned and was discovered independently by Specht [156] and Good [62]. Although this strategy benefits from translating the problem to an eigenvalue problem for which very robust and fast algorithms exist, it is preferred to work with low-degree matrices arising from low-degree polynomials. Following the key idea in [18], the roots command in Chebfun splits the polynomial recursively into smaller subintervals until pieces of degree less or equal to 100 are obtained. Then, the roots of each piece are computed as described above and collected with the roots of all the other pieces. For piecewise smooth chebfuns the strategy is the same, computing the roots of every smooth part and bookkeeping possible roots at break points.

As happens with other functions, the operation of `sum` (integral computed with Clenshaw–Curtis quadrature), `cumsum` (indefinite integral) and `diff` (derivative) on a piecewise smooth chebfun are carried out by first applying the operations on the smooth parts. In the case of `sum`, the results are added, and for `cumsum` the indefinite integral is obtained by shifting each piece the appropriate amount.

As mentioned before, unit impulses and their derivatives are stored in the matrix of the field `imps`. The value of the "jump" associated with the ith derivative of the unit impulse, $i \geq 0$, at jth break point between smooth pieces is stored in `imps(i+1,j)`, and the commands `diff` and `cumsum` introduce and remove a row in `imps`, respectively.

3.5 Discussion

We have presented in this chapter the strategy that we have implemented in Chebfun for approximating piecewise smooth functions. The crucial information required for their construction is the location of the singularities, and we have presented the three ways in which this is done in Chebfun: explicitly (specifying the locations in the constructor), implicitly (by the knowledge of functions that introduce discontinuities), and automatically (with the subdivision/edge-detection algorithm).

The idea of splitting the polynomial approximating the function into smaller intervals is a powerful one, found in the `splitting on` mode of Chebfun, or its powerful rootfinder command `roots`. In both cases a recursion is set, with the condition that if a polynomial has a degree higher than a certain value, the interval is split in two. A fundamental improvement to the splitting strategy for the construction of piecewise smooth chebfuns was the use of an additional procedure that attempts to locate the exact location of the discontinuity by comparing the possible growth of estimates of the derivatives. Finite differences provide simple estimates that have proven to be reliable.

An important aspect of Chebfun is that it assumes that the evaluation of the underlying function is computationally inexpensive, and we have presented examples in which thousands of samples are taken before resolving the approximation to machine precision.

We have given only a glimpse of the specific aspect of construction in Chebfun. However the project encompasses many developments that we have not even mentioned here; complete and up-to-date information can be found at the webpage. Closer to the subject of this thesis is the forthcoming book by Trefethen [164], in which many classical and modern topics in approximation theory are treated with the aid of Chebfun.

CHAPTER 4

A New Method for Rational Interpolation[1]

A few minutes later, strolling home on Fifth Avenue, I wondered if Dr. Heimlich, whose name is now so firmly placed in the national consciousness as the discoverer of the marvellous maneuver I had just seen performed, had any idea of how close he had once come to being scooped by three still utterly anonymous scientists who had worked for months on end in search of a cure for the same perilous mealtime trauma.

Woody Allen
Side Effects (1980)

Let $\mathcal{R}(m,n)$ be the family of rational functions of the form p/q, where p and q are polynomials in $\mathcal{P}(m)$ and $\mathcal{P}(n)$ respectively, and m and n are nonnegative integers. We always assume that a representation of r is in irreducible form, i.e. p and q have no common factors other than constants. In this chapter we consider the numerical solution of the following rational interpolation problem, also known as the *Cauchy interpolation problem* [26]: Given $\mathbf{x} = [x_0, \ldots, x_{m+n}]^\mathsf{T}$ and $\mathbf{f} = [f_0, \ldots, f_{m+n}]^\mathsf{T}$, both in $\mathbb{C}^{m+n+1}$, with $x_i \neq x_j$ if $i \neq j$, determine a function $r \in \mathcal{R}(m,n)$ such that

$$r(x_j) = \frac{p(x_j)}{q(x_j)} = f_j, \quad \text{for } j = 0, \ldots, m+n. \tag{4.1}$$

As in the previous chapter, we refer to $\mathbf{x}$ as the *set of nodes* or the *grid*, and to a rational function in $\mathcal{R}(m,n)$ as being of *type* $[m/n]$. Moreover, we use the notation $N = m + n$.

[1]The material in this chapter is largely adapted from R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., submitted [123]. This work began in the summer of 2009 when Pachón developed a new algorithm for rational interpolation in Chebyshev points. During the fall of 2009, Gonnet and van Deun were at Oxford, the first as a postdoc and the second in a four-month visit. Soon began a collaboration between the three of them that took several months, and the method presented in this chapter is the result of that joint collaboration. The paper was written by Pachón.

The advantages of rational over polynomial approximation come at the cost of increasing the complexity of the problem, since the construction of the former, in contrast with that of the latter, is nonlinear unless the poles are prescribed. Particular difficulties, such as the possible nonexistence of a solution, poles that may arise in the interpolation interval, or degeneracy of the rational function, are well known to affect the Cauchy interpolation problem [182]. These issues must be handled by an algorithm that computes rational interpolants, and there is a substantial literature on this subject. We will not attempt to present a survey of the methods which have been proposed in the past, but refer the reader to the expositions in [2, Sec. 7.1] and [113] and the references therein.

In this chapter we present a novel solution of the Cauchy interpolation problem. We are motivated by the simplicity and robustness that can be achieved in the polynomial case and we consider that a similar reliability would be desired when using rational functions. The algorithm is simple and useful for practical experimentation. A fundamental aspect of this method is that the basis of $\mathcal{P}(N)$ used to represent p and q depends on the grid, and the main idea is the following. First, construct the matrix $\hat{Z} \in \mathbb{C}^{(N+1)\times(N+1)}$ that transforms the coefficients $\hat{\boldsymbol{\beta}} \in \mathbb{C}^{N+1}$ of an arbitrary polynomial $\hat{q} \in \mathcal{P}(N)$ (in a basis to be specified) into the coefficients $\hat{\boldsymbol{\alpha}} \in \mathbb{C}^{N+1}$ of a polynomial $\hat{p} \in \mathcal{P}(N)$ such that $\hat{p}(x_i) = \hat{q}(x_j)f_j$. Second, constrain $\hat{Z}$ to obtain a linear system Z such that the associated numerator and denominator polynomials, p and q, are in $\mathcal{P}(m)$ and $\mathcal{P}(n)$ respectively. Notice that the resulting polynomials satisfy the conditions

$$p(x_j) = f_j q(x_j), \quad \text{for } j = 0, \dots, N. \tag{4.2}$$

If q is such that $q(x_j) \neq 0$, $j = 0, \dots, N$, the function $r = p/q \in \mathcal{R}(m,n)$ satisfies the original conditions (4.1). Modifying the nonlinear problem to a linear one is a common strategy for computing and analysing rational approximants. In the remaining of this chapter we will focus on the solution of (4.2) instead of (4.1), however the connection between the original problem and the modified one is important and we will say more about that on Chapter 5.

In Section 4.1 we introduce a method for rational interpolation in roots of unity. In this case the method is particularly simple and can be efficiently implemented using the FFT. In Section 4.2 we present the new method for rational interpolation in a general grid, taking particular interest in the cases of Chebyshev and equidistant points.

The Cauchy interpolation problem has been studied for almost two centuries, and it has a rich history. The methods introduced in this thesis have a connection with the classic algorithm by Jacobi [80] and in Section 4.3 we present some of the many relations they have with other works, such as the barycentric rational interpolation method proposed by Schneider and Werner [152] and continued by Berrut and Mittelmann [13], and the extension of Jacobi's algorithm developed by Eğecioğlu and Koç [39] based on orthogonal polynomials.

4.1 Rational interpolation in roots of unity

In this section we present a solution of the rational interpolation problem when the grid is the vector $\mathbf{z} = [z_0, \dots, z_N]^\mathsf{T}$ of $(N+1)$-st roots of unity (2.3). The derivation for this particular case in Section 4.1.1 is useful to highlight the ideas of the method for arbitrary grids that we will present in Section 4.2. An implementation using FFTs and the barycentric interpolation formula is presented in Section 4.1.2.

4.1.1 Derivation of the method

We begin by presenting the construction of a matrix $\hat{Z}$ that maps the coefficients $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0, \dots, \hat{\beta}_N]^\mathsf{T}$ of $\hat{q} \in \mathcal{P}(N)$, expressed as

$$\hat{q}(z) = \sum_{j=0}^{N} \hat{\beta}_j z^j,$$

into the coefficients $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_0, \dots, \hat{\alpha}_N]^\mathsf{T}$ of $\hat{p} \in \mathcal{P}(N)$, expressed as

$$\hat{p}(z) = \sum_{j=0}^{N} \hat{\alpha}_j z^j,$$

whose values at $\mathbf{z}$ are given by the entrywise product of $\mathbf{f}$ and $\hat{\mathbf{q}}$, that is,

$$[\hat{p}(z_0), \dots, \hat{p}(z_N)]^\mathsf{T} = [f_0 \hat{q}(z_0), \dots, f_N \hat{q}(z_N)]^\mathsf{T},$$

or $\hat{\mathbf{p}} = \mathbf{f} \cdot \hat{\mathbf{q}}$, where $\hat{\mathbf{p}} = \hat{p}(\mathbf{z})$ and $\hat{\mathbf{q}} = \hat{q}(\mathbf{z})$ in this section.

As in Section 2.1.1, A denotes the Vandermonde matrix $A = [\mathbf{z}^0 | \cdots | \mathbf{z}^N] \in \mathbb{C}^{(N+1)\times(N+1)}$. Denote by Φ the diagonal matrix of size $(N+1) \times (N+1)$ with (j,j) entry $\Phi_j = f_j$. From the discrete orthogonality property for roots of unity (2.4), it follows that

$$(A^* \Phi A)\hat{\boldsymbol{\beta}} = A^* \Phi \hat{\mathbf{q}} = A^*(\mathbf{f} \cdot \hat{\mathbf{q}}) = A^* \hat{\mathbf{p}} = A^* A \hat{\boldsymbol{\alpha}} = (N+1)\hat{\boldsymbol{\alpha}}.$$

Thus, the matrix

$$\hat{Z}_\mathbf{f} = \hat{Z} := A^* \Phi A \in \mathbb{C}^{(N+1)\times(N+1)}, \tag{4.3}$$

transforms the coefficients $\hat{\boldsymbol{\beta}}$ into the values $\hat{\mathbf{q}}$, computes $\hat{\mathbf{q}} \cdot \mathbf{f}$, and transforms these values back to the coefficients $\hat{\boldsymbol{\alpha}}$ times a constant factor. Notice that $\hat{Z}$ depends on $\mathbf{f}$ but we drop the subscript to simplify the notation. A schematic representation of $\hat{Z}$ is

$$\hat{Z} = \left[\begin{array}{c} (\mathbf{z}^0)^* \\ \hline \vdots \\ \hline (\mathbf{z}^N)^* \end{array}\right] \begin{bmatrix} f_0 & & \\ & \ddots & \\ & & f_N \end{bmatrix} \left[\begin{array}{c|c|c} \mathbf{z}^0 & \cdots & \mathbf{z}^N \end{array}\right]. \tag{4.4}$$

If $\hat{q}(x_j) \neq 0$, $j = 0, \dots, N$, then the rational function $\hat{r} = \hat{p}/\hat{q}$ satisfies the interpolation condition (4.1) for $\mathbf{z}$ and $\mathbf{f}$. However, $\hat{r}$ is in general a function of type $[N/N]$ and it is necessary to constrain the numerator and denominator to be polynomials in $\mathcal{P}(m)$ and $\mathcal{P}(n)$ respectively.

To constrain the degree of the denominator, we restrict the row space of $\hat{Z}$ to $\mathbb{C}^{n+1}$ by considering coefficient vectors of the form

$$\hat{\boldsymbol{\beta}} = [\beta_0, \dots, \beta_n, \underbrace{0, \dots, 0}_{m \text{ zeros}}]^\mathsf{T}, \tag{4.5}$$

for which it follows that

$$(N+1)\hat{\boldsymbol{\alpha}} = \hat{Z}\hat{\boldsymbol{\beta}} = A^*\Phi \left[\begin{array}{c|c|c} & & \\ \mathbf{z}^0 & \cdots & \mathbf{z}^n \\ & & \end{array}\right] \begin{bmatrix} \beta_0 \\ \vdots \\ \beta_n \end{bmatrix}. \tag{4.6}$$

Hence, the last m columns of the rightmost matrix A in (4.3) can be removed when computing the first $n+1$ coefficients of q. To constrain the degree of the numerator, we need to obtain conditions that guarantee that its coefficients are of the form

$$\hat{\boldsymbol{\alpha}} = [\alpha_0, \dots, \alpha_m, \underbrace{0, \dots, 0}_{n \text{ zeros}}]^\mathsf{T}.$$

For such $\hat{\boldsymbol{\alpha}}$, an arbitrary vector $\hat{\boldsymbol{\beta}}$ satisfies $\hat{Z}\hat{\boldsymbol{\beta}} = (N+1)\hat{\boldsymbol{\alpha}}$ if and only if it also satisfies the reduced system

$$\begin{bmatrix} (\mathbf{z}^{m+1})^* \\ \hline \vdots \\ \hline (\mathbf{z}^N)^* \end{bmatrix} \Phi A \hat{\boldsymbol{\beta}} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{4.7}$$

Thus, to restrict the degree of the numerator, we modify the system (4.3) by replacing leftmost matrix A^*, in the right-hand side product, with its last n rows.

Combining (4.6) and (4.7) we obtain the matrix (recall notation in page 10)

$$Z = A^*_{m+1:N}\Phi A_{0:n} \in \mathbb{C}^{n\times(n+1)}, \tag{4.8}$$

and it follows that a vector $\boldsymbol{\beta} = [\beta_0, \dots, \beta_n]^\mathsf{T}$ such that

$$Z\boldsymbol{\beta} = 0,$$

i.e. $\boldsymbol{\beta}$ is in the kernel of Z, and a vector $\boldsymbol{\alpha} = [\alpha_0, \dots, \alpha_m]^\mathsf{T}$ given by $A^*_{0:m}\Phi A_{0:n}\boldsymbol{\beta} = (N+1)\boldsymbol{\alpha}$, determine the polynomials $p = \sum_{j=0}^{m} \alpha_j z^j$ and $q = \sum_{j=0}^{n} \beta_j z^j$ that satisfy (4.2) for a grid of roots of unity.

4.1.2 Implementation with FFTs and barycentric formula

We mentioned in Section 2.1.1 that the matrix-vector multiplication $A^*\mathbf{x}$ involves the $(N+1)$-point DFT matrix and can be quickly computed by the fast Fourier transform in $\mathcal{O}(N\log N)$ operations instead of the usual $\mathcal{O}(N^2)$. Similarly, the matrix-vector product $A\tilde{\mathbf{x}}$ produces $\mathbf{x}$, the inverse DFT of the vector $\tilde{\mathbf{x}}$, and it can be efficiently computed by the inverse fast Fourier transform. Since

$$\hat{Z} = A^*\Phi A = A^*(A^*(\Phi^*))^*,$$

we can obtain the matrix $\hat{Z}$ by forming the matrix D whose columns are the FFT of each column of Φ^*, and then computing the FFT of D^*. The matrix Z is computed in the same way but requires the deletion of the last m rows of D and the first $m+1$ rows of the FFT of D^*. In Matlab, if `fk` is a vector of length $N+1$ with the values to be interpolated at roots of unity, the following three lines compute the matrix Z:

```
>> D  = fft( diag(conj(fk)) );
>> DD = fft( D(1:n+1,:)' );
>> Z  = DD(m+2:N+1,:);
```

From an element $\boldsymbol{\beta}$ in the kernel of Z, obtained for example from the Singular Value Decomposition [168], we obtain the $m+1$ coefficients $\boldsymbol{\alpha}$ of the numerator p by computing $A^*_{0:m}\Phi A_{0:n}\boldsymbol{\beta} = (N+1)\boldsymbol{\alpha}$. The matrix-vector product $A_{0:n}\boldsymbol{\beta} = \mathbf{q}$ computes the values of the denominator polynomial q in roots of unity, i.e. $\mathbf{q} = q(\mathbf{z})$. Since

$$A_{0:n}\boldsymbol{\beta} = A\hat{\boldsymbol{\beta}},$$

where $\hat{\boldsymbol{\beta}}$ is given by (4.5), we can compute $\mathbf{q}$ more efficiently for large N by taking the $(N+1)$-point inverse FFT of $\boldsymbol{\beta}$. The vector $\boldsymbol{\alpha}$ corresponds to the first $m+1$ entries of the FFT of $\mathbf{f}\cdot\mathbf{q}$ (the rest of the entries, by construction, are zero).

However, for the evaluation of a rational interpolant $\hat{r}$ at a single point x, a method other than explicitly evaluating the quotient $\hat{p}(x)/\hat{q}(x)$ is preferable: The *rational barycentric formula*

$$\hat{r}(x) = \frac{\displaystyle\sum_{j=0}^{N}\frac{u_j}{x-x_j}f_j}{\displaystyle\sum_{j=0}^{N}\frac{u_j}{x-x_j}}, \tag{4.9}$$

where $u_j = w_j\hat{q}(x_j)$ are the rational barycentric weights, where w_j are the polynomial barycentric weights w_j defined by (2.15). The rational barycentric formula is numerically stable, as shown by Salazar Celis [148]. If $\hat{q}(\mathbf{x})$ is a constant vector, then (4.9) collapses to the barycentric formula for polynomial interpolation (2.21). For an arbitrary vector

$\hat{q}(\mathbf{x}) \in \mathbb{C}^{N+1}$ with entries all different from zero, (4.9) defines a function $\hat{r}$ in $\mathcal{R}(N,N)$ that interpolates $\mathbf{f}$ in $\mathbf{x}$. The underlying polynomials $\hat{p}$ and $\hat{q}$ in $\mathcal{P}(N)$ for the numerator and denominator are made evident when writing them as Lagrange interpolants of the values $\hat{q}(\mathbf{x}) \cdot \mathbf{f}$ and $\hat{q}(\mathbf{x})$ respectively:

$$\hat{p}(x) = \ell(x) \sum_{j=0}^{N} \frac{w_j \hat{q}(x_j)}{x - x_j} f_j, \quad \hat{q}(x) = \ell(x) \sum_{j=0}^{N} \frac{w_j \hat{q}(x_j)}{x - x_j}. \tag{4.10}$$

It is evident from (4.10) that $\hat{r}$ always interpolates $\mathbf{f}$ exactly in $\mathbf{x}$. When $x = x_i$ the evaluation of r should be f_i instead of attempting to compute the value from formula (4.9), which is not defined in that case. Conversely, using the same argument it follows that any interpolant of type $[N/N]$ can be written in the barycentric form (4.9) for some rational barycentric weights u_j [12].

The arbitrary choice of $\hat{\mathbf{q}}$ gives $N+1$ degrees of freedom that can be used to impose additional characteristics on the rational interpolant $\hat{r}$. For example, for the Cauchy interpolation problem we have seen that these values are chosen in such a way that the rational function is of type $[m/n]$, as noticed first, in the barycentric setting, by Schneider and Werner [152]. Another example are the interpolants of Berrut [10] and Floater and Hormann [48], for which additional conditions are imposed to guarantee that $\hat{r} \in \mathcal{R}(N,N)$ is free of poles while keeping high rates of convergence as $N \to \infty$. For a survey of rational barycentric interpolation, see [12].

The code in Figure 4.1 presents a Matlab implementation of the ideas described so far. To obtain an element of the kernel of Z we compute its last right singular vector obtained from the Matlab command `svd`, which takes no more than a couple of seconds on a workstation even when n is in the hundreds. The singular values of Z provide information on the numerical aspects of the rational interpolation problem as we will show in Section 5.2.

```
function rh = ratinterproots(fh,m,n)

N = m+n;                           % N+1 = # of nodes
z = exp(2i*pi*(0:N)/(N+1)).';      % roots of unity
f = fh(z);                         % function values
D = fft(diag(conj(f)));            % compute A'(Phi')
Z = fft(D(1:n+1,:)');              % compute A'(A'(Phi'))'
[u,s,v] = svd(Z(m+2:N+1,:));       % svd of syst w size nx(n+1)
q = ifft(v(:,end),N+1);            % values of q at roots of unity
rh = @(x) bary(x,f,z,q.*z);        % rat. bary. interpolation
```

Figure 4.1: Matlab code for the rational interpolation algorithm in roots of unity. The input arguments are a function handle `fh` that specifies the values to be interpolated at `z`, and the degrees `m` and `n`. The output argument is a function handle `rh` for the evaluation of the rational interpolant of type $[m/n]$. If the vector `q` does not have zeros, the rational interpolant satisfies (4.1).

The only command that is not part of standard Matlab is `bary`, which is included in Chebfun to evaluate the barycentric formula with arbitrary weights. In general,

```
 r_x = bary(x,fvals,xk,uk)
```

interpolates the values `fvals` at nodes `xk` in the point `x` using the weights `uk`[2].

The following lines compute the interpolant in $\mathcal{R}(45,4)$ of the function $f(z) = \log(2-z)\sqrt{z+2}/(1-16z^4)$ in 50 roots of unity and evaluate it on a grid of 200 points. As expected, the rational interpolant is a good approximation of the function in the unit disk [147]:

```
>> fh = @(z) log(2-z).*sqrt(z+2)./(1-16*z.^4);
>> tic, rh = ratinterproots(fh,50,4); toc
Elapsed time is 0.000590 seconds.
>> zz = exp(1i*linspace(0,2*pi,200));
>> error_zz = norm(fh(zz)-rh(zz),inf);
error_zz = 1.792609524364659e-16
```

For large N, the total computational cost is $\mathcal{O}(N^2 \log N)$ operations to compute the matrix Z in (4.8), $\mathcal{O}(n^3)$ operations to obtain its kernel vector, $\mathcal{O}(N \log N)$ operations to compute the values in $\mathbf{q}$, and $\mathcal{O}(N)$ operations to evaluate the barycentric formula (4.9).

4.2 Rational interpolation in other grids

In this section we present a solution of the Cauchy interpolation problem for an arbitrary grid $\mathbf{x} \in \mathbb{C}^{N+1}$. The basis that we use to represent the numerator and denominator is a family $\{\phi_j\}$, $j = 0, \ldots, N$, where $\phi_j \in \mathcal{P}(j)$, and orthogonal with respect to the inner product

$$\langle f, g \rangle_{\mathbf{x}} = \sum_{k=0}^{N} f(x_k)\overline{g(x_k)}, \tag{4.11}$$

that is,

$$\langle \phi_j, \phi_k \rangle_{\mathbf{x}} = \begin{cases} \kappa_j > 0, & \text{if } j = k, \\ 0, & \text{if } j \neq k. \end{cases} \tag{4.12}$$

Defining the Vandermonde-type matrix $C = [\phi_0(\mathbf{x})|\cdots|\phi_N(\mathbf{x})] \in \mathbb{C}^{(N+1)\times(N+1)}$ and $\hat{\boldsymbol{\kappa}} = [\kappa_0, \ldots, \kappa_N]^\mathsf{T}$, (4.12) can be written as $C^*C = \operatorname{diag}(\hat{\boldsymbol{\kappa}})$. The following theorem summarises the method presented in this chapter.

[2]The `bary` command can be replaced by the following function in Matlab, written by Pedro Gonnet:

```
function y = bary (x, fx, xi, w )
y = zeros(size(x)); fxw = fx(:).*w(:);
for i=1:length(x),
    dxinv = 1.0 ./ ( x(i) - xi(:)  ); ind = find(  isfinite(dxinv) );
    if length(ind)>0, y(i)=fx(ind); else, y(i)=(fxw.'* dxinv)/(w(:).'* dxinv); end
end
```

Theorem 4.1. *If* $\boldsymbol{\alpha} = [\alpha_0, \dots, \alpha_m]^\top$ *and* $\boldsymbol{\beta} = [\beta_0, \dots, \beta_n]^\top$ *are such that*

$$\boldsymbol{\beta} \in \ker(C^*_{m+1:N} \Phi C_{0:n}) \quad \textit{and} \quad C^*_{0:m} \Phi C_{0:n} \boldsymbol{\beta} = \boldsymbol{\kappa} \cdot \boldsymbol{\alpha},$$

where $\boldsymbol{\kappa} = [\kappa_0, \dots, \kappa_m]^\top$*, then the polynomials*

$$p(x) = \sum_{j=0}^{m} \alpha_j \phi_j(x), \quad \textit{and} \quad q(x) = \sum_{j=0}^{n} \beta_j \phi_j(x),$$

satisfy the conditions (4.2)*. If* $q(x_j) \neq 0$*, the function* $r = p/q \in \mathcal{R}(m, n)$ *satisfies the conditions* (4.1)*.*

Proof. Let $\hat{p}(x) = \sum_{j=0}^{N} \hat{\alpha}_j \phi_j(x)$ and $\hat{q}(x) = \sum_{j=0}^{N} \hat{\beta}_j \phi_j(x)$, such that $\hat{\mathbf{p}} = \mathbf{f} \cdot \hat{\mathbf{q}}$, where $\hat{\mathbf{p}} = \hat{p}(\mathbf{x})$ and $\hat{\mathbf{q}} = \hat{q}(\mathbf{x})$. Then it follows that

$$(C^* \Phi C)\hat{\boldsymbol{\beta}} = C^* \Phi \hat{\mathbf{q}} = C^*(\mathbf{f} \cdot \hat{\mathbf{q}}) = C^* \hat{\mathbf{p}} = C^* C \hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\kappa}} \cdot \hat{\boldsymbol{\alpha}}.$$

Equations (4.6)–(4.8) in Section 4.1.1 apply for the matrix $\hat{Z} = C^* \Phi C$, with A replaced by C, from which the result of the theorem follows. $\square$

In the remainder of the thesis we denote $C^*_{m+1:N} \Phi C_{0:n} \in \mathbb{C}^{n \times (n+1)}$ in Theorem 4.1 by Z. We use Theorem 4.1 to obtain a solution of the rational interpolation problem on various specific grids. For each of them we establish the family of polynomials $\{\phi_j\}$ orthogonal with respect to (4.11). These polynomials, in most cases, can be defined, from certain initial conditions, by a three-term recurrence relation of the form

$$\phi_{k+1}(x) = a_k(x - b_k)\phi_k(x) - c_k \phi_{k-1}(x). \tag{4.13}$$

4.2.1 Chebyshev points

The following discrete orthogonality property is satisfied for Chebyshev polynomials [104, Sec. 4.6.1]:

$$\langle T_j, T_k \rangle_{\mathbf{y}^{(1)}} = \begin{cases} \kappa_j & \text{if } j = k, \\ 0 & \text{if } j \neq k, \end{cases} \tag{4.14}$$

where $\mathbf{y}^{(1)}$ is a grid of $(N+1)$ Chebyshev points of the first kind given by (2.8), with $\kappa_0 = N+1$ and $\kappa_j = \frac{1}{2}(N+1)$, $j = 1, \dots, N$. Thus, the matrix C above can be constructed with Chebyshev polynomials and the rational interpolant in $\mathbf{y}^{(1)}$ is established from Theorem 4.1. We present in Figure 4.2 an implementation of the method for rational interpolation in Chebyshev points of the first kind[3].

[3]The `idct` command for the inverse DCT can be replaced by the following five lines (where `y` and `x` are the input and output vectors respectively), written by Pedro Gonnet:

```
n = size(y,1); w = (exp(1i*(0:n-1)*pi/(2*n)) * sqrt(2*n)).';
if mod(n,2) == 1 ||  isreal(y), w(1) = w(1)*sqrt(2);
x = ifft([diag(w)*y;zeros(1,size(y,2));-1i*diag(w(2:n))*y(n:-1:2,:)]);
x = x(1:n,:); else w(1) = w(1)/sqrt(2); x([1:2:n,n:-2:2],:)  = ifft(diag(w)*y); end
if isreal(y), x = real(x); end
```

```
function rh = ratinterpcheb1(fh,m,n)

N = m+n;                                         % N+1 = # of nodes
y1 = cos((2*(0:N)+1)*pi/(2*N+2))';               % Cheb pts of 1st kind
f = fh(y1);                                      % function values
D = dct(diag(f));                                % compute C'(Phi')
Z = dct(D(1:n+1,:)');                            % compute C'(C'(Phi'))'
[u,s,v] = svd(Z(m+2:N+1,:));                     % svd of syst w size nx(n+1)
q = idct(v(:,end),N+1);                          % values of q at Cheb pts
w = (-1).^(0:N)'.*sin((2*(0:N)+1)*pi/(2*N+2))';  % barycentric weights, see (2.18)
rh = @(x) bary(x,f,y1,q.*w);                     % rat. bary. interpolation
```

Figure 4.2: Matlab code for the rational interpolation algorithm in Chebyshev points of the first kind. Input and output arguments are similar to those in Figure 4.1.

The following lines compute an interpolant in $\mathcal{R}(12,12)$ of the function $f(x) = (1.5 - \cos(5x))^{-1}$ in 25 Chebyshev points of the first kind. The error of the rational interpolant in 200 equispaced points is of order 10^{-15}:

```
>> fh = @(x) 1./(1.5-cos(5*x));
>> tic, rh = ratinterpcheb1(fh,12,12); toc
Elapsed time is 0.000853 seconds.
>> xx = linspace(-1,1,200);
>> error_xx = norm(fh(xx)-rh(xx),inf)
error_xx = 1.332267629550188e-15
```

We can also obtain an efficient method for rational interpolation in Chebyshev points of the second kind, given by (2.9). Orthogonal polynomials with respect to $\langle\cdot,\cdot\rangle_{\mathbf{y}^{(2)}}$ have been recently derived by Eisinberg and Fedele [42]. They are

$$\phi_0(x) = N-1, \quad \phi_1(x) = Nx, \quad \text{and } \phi_2(x) = (N+1)x^2 - \frac{N+2}{2},$$

and higher degree polynomials are defined by the three-term recurrence (4.13) with coefficients given by

$$a_k = \frac{N+k}{N+k-1}, \quad b_k = 0, \quad \text{and} \quad c_k = \frac{N+k+1}{4(N+k-1)},$$

for $k = 2, \ldots, N-1$. For $\langle\cdot,\cdot\rangle_{\mathbf{y}^{(2)}}$, they satisfy a similar condition as (4.14), with the values $\kappa_j = \langle\phi_j,\phi_j\rangle_{\mathbf{y}^{(2)}}$ given exactly by

$$\kappa_j = \begin{cases} (N+1)(N-1)^2, & \text{if } j = 0, \\ \dfrac{N(N+j+1)(N+j-1)}{2^{2j-1}}, & \text{if } 0 < j < N, \\ \dfrac{N^2(2N-1)}{2^{2N-3}}, & \text{if } j = N. \end{cases}$$

It follows from Theorem 4.1 that a rational interpolant for $\mathbf{y}^{(2)}$ can be constructed from these polynomials. The method, however, can be slightly modified in this case to make it simpler by using Chebyshev polynomials instead. In particular, Chebyshev polynomials are orthogonal with respect to the inner product [104, Sec. 4.6.1]

$$(f, g)_{\mathbf{y}^{(2)}} = \sum_{k=0}^{N}{}'' f\big(y_k^{(2)}\big)\overline{g\big(y_k^{(2)}\big)}, \tag{4.15}$$

that is, similarly to (4.14), they satisfy

$$(T_j, T_k)_{\mathbf{y}^{(2)}} = \begin{cases} \kappa_j, & \text{if } j = k, \\ 0, & \text{if } j \neq k, \end{cases} \tag{4.16}$$

with $\kappa_0 = \kappa_N = N$ and $\kappa_j = \frac{1}{2}N$, $j = 1, \ldots, N-1$, i.e. if C is defined by Chebyshev polynomials evaluated on $\mathbf{y}^{(2)}$, we have that $C^\mathsf{T} I'' C = \operatorname{diag}(\hat{\boldsymbol{\kappa}})$, where I'' is the $(N+1) \times (N+1)$ identity matrix with the top-left and bottom-right entries halved (recall notation in page 10).

If $\hat{p}(x) = \sum_{j=0}^{N} \hat{\alpha}_j T_j(x)$ and $\hat{q}(x) = \sum_{j=0}^{N} \hat{\beta}_j T_j(x)$ such that $\hat{\mathbf{p}} = \hat{\mathbf{q}} \cdot \mathbf{f}$, where $\hat{\mathbf{p}} = \hat{p}(\mathbf{y}^{(2)})$ and $\hat{\mathbf{q}} = \hat{q}(\mathbf{y}^{(2)})$, then, similarly as in the proof of Theorem 4.1, we have that

$$(C^\mathsf{T}\Phi'' C)\hat{\boldsymbol{\beta}} = C^\mathsf{T}\Phi''\hat{\mathbf{q}} = C^\mathsf{T} I''(\mathbf{f} \cdot \hat{\mathbf{q}}) = C^\mathsf{T} I''\hat{\mathbf{p}} = C^\mathsf{T} I'' C\hat{\boldsymbol{\alpha}} = \hat{\boldsymbol{\kappa}} \cdot \hat{\boldsymbol{\alpha}}. \tag{4.17}$$

Notice that an alternative derivation of (4.17) can be obtained by applying the Lobatto–Markov quadrature rule to the integral formula $\hat{\alpha}_j = \int_{-1}^{1} \hat{p}(x)(1-x^2)^{-1}dx$ of the coefficient $\hat{\alpha}_j$ of the polynomial $\hat{p}$ [140, p. 151].

Following the same reasoning as in Theorem 4.1, the coefficients $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ of p and q are given by

$$\boldsymbol{\beta} \in \ker(C_{m+1:N}^\mathsf{T}\Phi'' C_{0:n}) \quad \text{and} \quad C_{0:m}^\mathsf{T}\Phi'' C_{0:n}\boldsymbol{\beta} = \boldsymbol{\kappa} \cdot \boldsymbol{\alpha},$$

For the implementation with the barycentric formula (4.9), the rational barycentric weights u_j are computed from the polynomial barycentric weights in (2.19), normalised as follows:

$$w_j = (-1)^j \delta_j, \quad \delta_j = \begin{cases} 1/2, & j = 0 \text{ or } j = N, \\ 1, & \text{otherwise.} \end{cases} \tag{4.18}$$

In Figure 4.3 we present a code for rational interpolation in Chebyshev points of the second kind using the modified method. In this case we explicitly form the Vandermonde-type matrix C by the three-term recurrence for Chebyshev polynomials.

4.2.2 Equidistant points

For the set $\mathbf{t}$ of $N+1$ equidistant nodes in $[-1, 1]$ the family $\{G_j\}$ of orthogonal polynomials with respect to $\langle \cdot, \cdot \rangle_{\mathbf{t}}$, required in Theorem 4.1, are the *Gram polynomials*

$$G_{-1} = 0, \quad G_0(x) = (N+1)^{-1/2},$$

```
function rh = ratinterpcheb2(fh,m,n)

N = m+n;                                        % N+1 = # of nodes
y2 = cos((0:N)*pi/N)';                          % Cheb pts of 2nd kind
f = fh(y2);                                     % function values
C(:,1) = ones(N+1,1); C(:,2) = y2;              % Vandermonde-type matrix
for k = 2:N,
    C(:,k+1) = 2*y2.*C(:,k) - C(:,k-1);         % 3-term recurrence
end
half = [1/2; ones(N-1,1);1/2];
Z = C(:,m+2:N+1).'*diag(half.*f)*C(:,1:n+1);  % modified matrix Z
[u,s,v] = svd(Z);                               % svd of syst w size nx(n+1)
q = C(:,1:n+1) * v(:,end);                      % values of q at Cheb pts
w = half.*(-1).^((0:N)');                       % barycentric weights
rh = @(x) bary(x,f,y2,q.*w);                    % rat. bary. interpolation
```

Figure 4.3: Matlab code for the rational interpolation algorithm for Chebyshev points of the second kind. Input and output arguments are similar to those in Figure 4.1. An implementation for this case could also use FFTs, but we present it in this way to illustrate the explicit construction of Z, required for general grids.

and higher degree polynomials defined by the three-term recurrence relation (4.13) with coefficients

$$a_k = \frac{N}{k+1}\left(\frac{4(k+1)^2-1}{(N+1)^2-(k+1)^2}\right)^{1/2}, \quad b_k = 0,$$

for $k = 0, \dots, N-1$, and $c_k = a_k/a_{k-1}$ for $k = 1, \dots, N-1$ (for more on Gram polynomials see [33, Sec. 4.4.4] and [41]). Specifically,

$$\langle G_j, G_k\rangle_{\mathbf{t}} = \begin{cases} 1, & \text{if } j = k, \\ 0, & \text{if } j \neq k. \end{cases}$$

For the implementation with formula (4.9), the simplified polynomial barycentric weights required to compute u_j are given by [14]

$$w_j = (-1)^j \binom{N}{j}. \tag{4.19}$$

Two difficulties arise for interpolation in equidistant points using the method presented in this work. First when N is large, Gram polynomials of degrees close to N have exponentially large oscillations near the endpoints, making them unsuitable for approximation of functions [3]. This introduces large numerical errors in the computation of the product $C^*_{m+1:N}\Phi C_{0:n}$ in Theorem 4.1, affecting the accuracy of the coefficient vector $\boldsymbol{\beta}$. Second, the weights (4.19) near the origin and near the endpoints vary by exponentially large factors. Thus, even if the computation of the values $q(t_j)$ were accurate, the interpolation with formula (4.9) would be unstable. Notice that rational interpolation methods that are known to perform better on equispaced points are not solutions of the Cauchy interpolation problem. For example, the interpolant of Floater and Hormann [48] constructs a rational function of type $[N/N]$ but not $[m/n]$.

These difficulties are not surprising, though, since it is known that equispaced grids are a poor choice for polynomial interpolation on an interval. A recent result by Platte, Trefethen and Kuijlaars, for example, shows that a stable algorithm for (linear or nonlinear) approximation from equispaced data cannot converge geometrically [128].

4.2.3 Arbitrary Grids

For an arbitrary grid $\mathbf{x}$ for which a family of orthogonal polynomials with respect to $\langle\cdot,\cdot\rangle_{\mathbf{x}}$ is not known explicitly, it is possible to obtain the associated matrix C by a numerical method [55, Chp. 2].

If $\langle\cdot,\cdot\rangle_{\mathbf{x}}$ satisfies the shift property $\langle xf, g\rangle_{\mathbf{x}} = \langle f, xg\rangle_{\mathbf{x}}$, then it follows that a three-term recurrence such as (4.13) can be established [55, Sec. 1.3], and the coefficients can be computed, for example, by the Stieltjes method [55, Sec. 2.2.3.1], where

$$a_k = 1, \quad b_k = \frac{\langle x\phi_k, \phi_k\rangle_{\mathbf{x}}}{\langle \phi_k, \phi_k\rangle_{\mathbf{x}}} \quad \text{and} \quad c_k = \frac{\langle \phi_k, \phi_k\rangle_{\mathbf{x}}}{\langle \phi_{k-1}, \phi_{k-1}\rangle_{\mathbf{x}}},$$

with initial conditions $\phi_{-1}(x) = 0$ and $\phi_0(x) = 1$, or by the orthogonal reduction method based on the Lanczos algorithm [55, Sec. 2.2.3.2]. See [53, Sec. 6] for a discussion of both methods, and the Matlab suite OPQ, written by Gautschi, for their implementation in the commands `stieltjes.m` and `lanczos.m` [54].

If the shift property is not satisfied, for example for grids in the complex plane that do not include all the conjugates of the grid points, the matrix C in Theorem 4.1 can be obtained by computing the QR decomposition of an $(N+1)\times(N+1)$ Vandermonde-type matrix, the columns of which are polynomials of an arbitrary basis evaluated at $\mathbf{x}$ [168]. Notice, however, that the number of operations of the QR decomposition is of order $\mathcal{O}(N^3)$.

4.3 History and Related Work

The first reference on the rational interpolation problem studied in this chapter was Cauchy's 1821 treatise on analysis for the École Royale Polytechnique [26]. In one of the appendices (*Note V. Sur la Formule de Lagrange relative à l'Interpolation*), he studies the generalisation of the Lagrange formula to rational functions. After showing that the condition $m + n = N$ guarantees the uniqueness of a solution, Cauchy derives an explicit though cumbersome formula for the rational interpolant[4]. See [113] for more on the history of this problem.

[4]Salzer [149] vividly describes the obscurity of this solution: "*Cauchy's formula is so rarely given, or even cited, that perhaps less than one person out of every thousand who can write down the Lagrange polynomial interpolation formula, can do the same for the former*".

A major contribution to the understanding of the rational interpolation problem was published by Jacobi in 1846 [80]. His solution is one of the main approaches for rational interpolation, another being the one based on continued fractions, of which we say nothing in this thesis. Jacobi's approach is as follows.

Suppose that $p \in \mathcal{P}(m)$, $q \in \mathcal{P}(n)$, and $p(\mathbf{x}) = q(\mathbf{x}) \cdot \mathbf{f}$. If we replace $\varphi(x)$ in the identity (2.24), valid for any polynomial of degree strictly less than N, by $p(x)\phi_j(x)$, where $\phi_j \in \mathcal{P}(j)$, $j = 0, \dots, n-1$, and express q as

$$q(x) = \sum_{k=0}^{n} \beta_k \psi_k(x),$$

then it follows that

$$\begin{aligned} 0 = \sum_{i=0}^{N} w_i p(x_i)\phi_j(x_i) &= \sum_{i=0}^{N} w_i f_i q(x_i)\phi_j(x_i) \\ &= \sum_{i=0}^{N} w_i f_i \phi_j(x_i) \sum_{k=0}^{n} \beta_k \psi_k(x_i). \end{aligned}$$

In matrix form, solving for $\boldsymbol{\beta}$ in the previous set of equations is equivalent to computing the kernel of the matrix

$$Z = \left[\begin{array}{c} \phi_0(\mathbf{x})^* \\ \hline \vdots \\ \hline \phi_{n-1}(\mathbf{x})^* \end{array}\right] \begin{bmatrix} f_0 w_0 & & \\ & \ddots & \\ & & f_N w_N \end{bmatrix} \left[\begin{array}{c|c|c} \psi_0(\mathbf{x}) & \cdots & \psi_n(\mathbf{x}) \end{array}\right]. \tag{4.20}$$

If the same basis $\{\phi_j\}$ is used to expand q, then

$$Z = C_{0:n-1}^{\mathsf{T}} \Phi \Omega C_{0:n}, \tag{4.21}$$

where Ω is the diagonal matrix of size $(N+1) \times (N+1)$ with (j,j) entry $\Omega_j = w_j$. Notice the similarity of (4.21) with the matrix of Theorem 4.1.

Consider the case in which

$$\begin{aligned} \phi_i(z) &= z^i, \quad i = 0, \dots, n-1, \\ \psi_j(z) &= z^j, \quad j = 0, \dots, n, \end{aligned}$$

and the grid is $\mathbf{z}$. From (2.20), and since $z_k^{(N-j+1)} = z_k^j$, it follows that (4.21) reduces to the same matrix Z obtained in Section 4.1 and given by (4.8). Similarly, consider the case in which the basis in (4.21) is fixed as

$$\begin{aligned} \phi_i(x) &= T_i(x), \quad i = 0, \dots, n-1, \\ \psi_j(x) &= T_j(x), \quad j = 0, \dots, n, \end{aligned}$$

and the grid is $\mathbf{y}^{(2)}$. From (4.18), and since $T_j(y_k^{(2)}) = (-1)^k T_{N-j}(y_k^{(2)})$, it follows that (4.21) reduces to the system $C^*_{m+1:N}\Phi'' C_{0:n}$, where the columns of C are Chebyshev polynomials evaluated at $\mathbf{y}^{(2)}$. This is the same system (4.17) resulting from the "modified method" of Section 4.2.1 for Chebyshev points of the second kind.

Algorithms based on a system like (4.21) have been proposed and rediscovered several times. The system (4.21) where $\{\phi_i\}$ and $\{\psi_i\}$ are sets of monomials have been used, for example, by Kronecker [85], Werner [180], Meinguet [113], and Graves-Morris [67].

More recently, Berrut and Mittelmann [13] proposed to solve Cauchy's interpolation problem by using monomials to define C, and directly computing $\mathbf{u} = \Omega C_{0:n}\boldsymbol{\beta} \in \mathbb{C}^{N+1}$, the weights of the rational barycentric formula (4.9), as an element in the kernel of

$$B = \begin{bmatrix} C^{\mathsf{T}}_{0:m-1} \\ C^{\mathsf{T}}_{0:n-1}\Phi \end{bmatrix} \in \mathbb{C}^{N\times(N+1)}.$$

The n conditions at the bottom of $B\mathbf{u} = 0$ are the same as those in (4.21). And the m conditions at the top are similarly derived from (2.24), using for that formula the polynomial $q(x)\phi_j(x)$ of degree strictly less than N, where $\phi_j \in \mathcal{P}(j)$, $j = 0, \ldots, m-1$. Reductions of the system to one of size $n \times (n+1)$ are studied in [13], [186] and [129]. Berrut has also proposed a modification of the previous system by using a barycentric representation of the rational interpolant that interpolates in grids with less than $N+1$ points [11, 12].

Instead of monomials, Opitz [122] and Schneider and Werner [152] used the Newton basis, and Predonzan [131] and Salzer [149] used the Lagrange basis.

Another method derived from (4.20), closer in spirit to the one introduced here, is the one proposed by Eğecioğlu and Koç [39]. Instead of fixing the basis, they used a Stieltjes procedure to obtain polynomials $\phi_j \in \mathcal{P}(j)$, $j = 0, \ldots, n$, orthogonal with respect to the bilinear form

$$\langle g, h\rangle_{\mathbf{f},\mathbf{x}} = \sum_{j=0}^{N} f_j w_j g(x_j) h(x_j). \tag{4.22}$$

At the kth step of the Stieltjes algorithm the polynomial ϕ_k in the matrix C of (4.21) is obtained from ϕ_{k-1} and ϕ_{k-2}. Since these polynomials are orthogonal with respect to (4.22), at the end of the nth step all the entries off the diagonal of Z are zero, and it follows that $\phi_n(x)$ corresponds to the denominator polynomial $q(x)$. This method is very fast, since there is no need for explicitly assembling Z or using an SVD algorithm to compute an element in its kernel. However, since the weight of the bilinear form (4.22) depends on the polynomial barycentric weights and the values $\mathbf{f}$, the method is sensitive not only to the grid distribution but also to the data values. In particular a special procedure must be used if any value f_j is zero. This method has been developed further in [60].

4.4 Discussion

We have presented a new algorithm for the solution of Cauchy's rational interpolation problem, summarised in Theorem 4.1. We established conditions that ensure that

- the interpolation conditions (4.2) are satisfied, and
- the polynomials p and q are of the appropriate degrees.

The first set of conditions are represented in the linear transformation $\hat{Z}$ that maps the coefficients of $q \in \mathcal{P}(N)$ to its values on the grid, computes then its entrywise product with the data $\mathbf{f}$, and finally take those values back as coefficients of $p \in \mathcal{P}(N)$. By construction, we enforce that $p(x_j) = q(x_j)f_j$, $j = 0, \ldots, N$. The second set of conditions can be viewed as the requirements to filter out the highest frequencies of p and q, and they are obtained by appropriately restricting the matrix $\hat{Z}$. The idea of describing the operator that connects the numerator and denominator of a rational interpolant as a series of transformations between the "coefficient-space" and "data-space" is new. It remains to determine whether a similar approach could be used to describe and understand other types of rational interpolants or rational approximants in general.

The new method relies on the use of polynomials that satisfy the discrete orthogonal property (4.12). Some recent research has focused on these polynomials, and for particular grids they are known explicitly. In Section 4.2.1 we modified the method for Chebyshev points of the second kind by requiring orthogonality with respect to (4.15) instead of (4.11). This suggests that the method can be modified and perhaps improved by introducing specific knowledge of polynomials orthogonal with respect to other inner products.

A question that we were not able to answer in this thesis is whether it is possible to simplify or completely avoid the computation of the SVD in the algorithm. Our concern is not that the cost of this operation is $\mathcal{O}(n^3)$, since we have rarely increased the degree of the denominator so much that the effect of this computation is noticeable. What is really behind our question is the possibility of extracting more information from the interpolation problem, which we think could be encrypted in the SVD. In the next chapter we will say more about our interest on the singular values of Z.

The barycentric formula (4.9) provides an efficient method for evaluating the rational interpolant. And for the specific cases of roots of unity and Chebyshev points of the first kind, the matrix Z can be constructed using the FFT algorithm. Notice, however, that for certain pairs (m, n) of the degrees of p and q, the explicit computation of the product $C^*_{m+1:N}\Phi C_{0:n}$ may be faster.

The identity (2.24) provides a family of methods for solving the rational interpolation problem by choosing different bases to represent p and q and computing the kernel of (4.21). We have found in the literature that the most common choices for this method have been the monomials, as used for the first time by Jacobi, the Lagrange basis, and the Newton basis. Eg̃ecioğlu and Koç also relied on that identity but used instead orthogonal bases that eliminate the computation of the associated linear system.

We started to work on rational interpolation in 2009, in connection with our work on the rational Remez algorithm. Following very closely the method of Berrut and Mittelmann [13], and thinking constantly in Chebyshev polynomials, we originally derived similar conditions as those presented in that paper but in which the effect of the system was to cancel out the highest Chebyshev coefficients of p and q, not their monomial coefficients. This leads essentially to the same problem of computing the kernel of (4.21) with Chebyshev polynomials as a fixed basis $\{\phi_j\}$, an option that has not been explored in the literature, but which might be a preferable choice in Chebfun.

Future work may aim at taking the new interpolation method in other directions. It would be interesting to study whether it can be used for other types of rational interpolation, as in the sense of Hermite or with prescribed poles, for example, or other types of rational approximants, as in the least squares sense.

CHAPTER 5

Approximation by Rational Functions[1]

Perinthia's astronomers are faced with a difficult choice. Either they must admit that all their calculations were wrong and their figures are unable to describe the heavens, or else they must reveal that the order of the gods is reflected exactly in the city of monsters.

Italo Calvino
Invisible Cities (1972)

The complexity of computing approximations by rational functions is justified by the possibility of improving certain aspects over the simpler polynomial case. Consider for example the extension of the Runge theorem to rational functions, which allows for a convergence result on more general regions of the complex plane than the ones considered in Theorem 2.2 [141, Thm. 13.6].

Theorem 5.1 (Runge). *Let $\mathcal{K}$ be a compact set in $\mathbb{C}$, and for each component C_j of $\mathbb{C}\backslash\mathcal{K}$ let a point $q_j \in C_j$ be given. Let f be an analytic function in a neighbourhood of $\mathcal{K}$. Then there exists a sequence of rational functions with (at most) poles in the set $\{q_j\}$ that converges uniformly to f on $\mathcal{K}$.*

Runge's theorem highlights the ability of rational functions to approximate in domains where the region of analyticity of the target function has "holes" due to isolated singularities. Other instances in which rational functions outperform polynomials include the approximation of functions with discontinuities (see, for example, the discussion on p. 74) or defined in unbounded domains [148].

The study of rational functions for almost two centuries has produced a long list of applications and connections with various classical topics in approximation theory such

[1]Parts of the material in this chapter are adapted from R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, SIAM J. Numer. Anal., submitted [123].

as extrapolation [64, 19], orthogonal polynomials [121], and quadrature [167]. Recent applications include the development of rational spectral collocation methods for differential equations [12, 162] and rational Krylov iteration methods for the computation of eigenvalues [74].

In this chapter we present theoretical results and observations on approximation by rational functions, with special emphasis on the interpolants constructed in Chapter 4, i.e. rational functions that solve the Cauchy interpolation problem. Recall that these approximants aim to interpolate on grids of $m + n + 1$ points with functions in $\mathcal{R}(m, n)$ the poles of which are not prescribed in advance.

The rational interpolants associated with a triangular sequence of points provide a sequence of approximations to a target function which can be studied in a similar way as we did with polynomial interpolants. Our themes here are those treated earlier in Sections 2.2–2.4: convergence to functions of different classes of smoothness, inference of the region of analyticity through inverse results, and alternative approximants obtained from the infinite series representation of the target function, i.e. Padé-type approximants.

The convergence analysis of rational interpolants must distinguish the ways in which the numerator and/or denominator degrees increase to infinity. For this purpose it is useful to mention the *rational interpolation table* associated with a target function f and a triangular sequence of points $\mathbf{X}$. This is defined as an infinite two-dimensional array

$$\begin{array}{cccc} [0/0] & [1/0] & [2/0] & \cdots \\ [0/1] & [1/1] & [2/1] & \cdots \\ [0/2] & [1/2] & [2/2] & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{array}$$

where the element in the m-th row and n-th column, for $m, n \geq 0$, corresponds to the rational function in irreducible form with numerator $p \in \mathcal{P}(m)$ and denominator $q \in \mathcal{P}(n)$ that satisfies the linearised interpolation conditions (4.2) on the $(m + n)$th row of $\mathbf{X}$. Typical analyses involve the study along rows (fixed degree of the denominator, increasing degree of the numerator), columns (fixed degree of the numerator, increasing degree of the denominator) and the diagonal (increasing degrees of numerator and denominator).

The literature on rational approximation is vast, and since the end of the 1970s an important part of the reseach has focused on Padé approximation (for some evidence of this, see Figure 5.1). The study of approximation by rational interpolants, on the other hand, seems peripheral and far from complete. The rational interpolant method that we introduced in Chapter 4 sparked our interest on this topic, and more precisely on its computational aspects.

We begin in Section 5.1 by studying the type of convergence results we should expect from rational interpolants as approximants. In particular we follow very closely the sort of

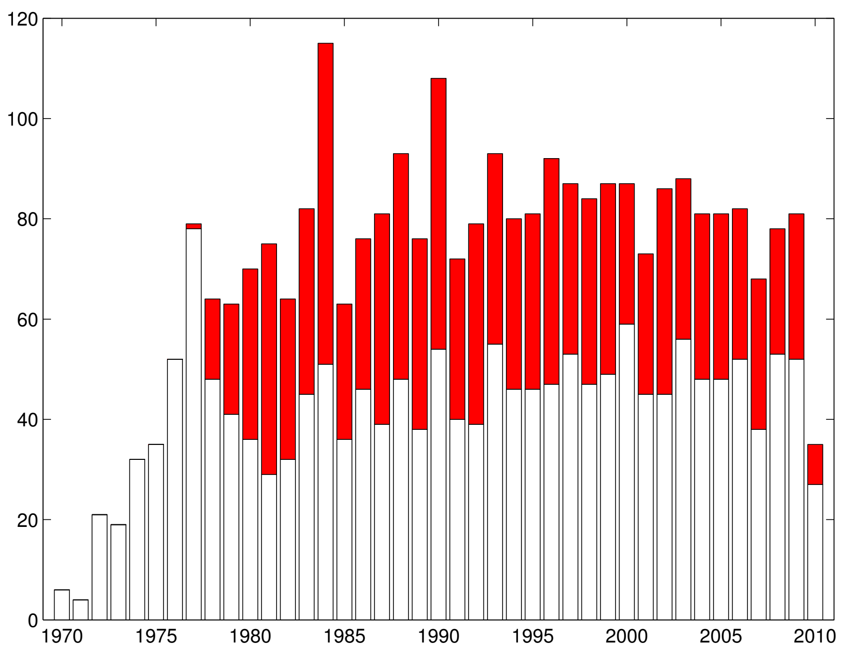

Figure 5.1: Number of publications in MathSciNet (`www.ams.org/mathscinet/`) on rational approximation 1970–2010 (as of September 2010). The white bars show the number of publications with MSC Primary/Secondary 41A20 (rational approximation) and the red bars show the number of publications with MSC Primary/Secondary 41A21 (Padé approximation) but not 41A20.

analyses that we carried out for polynomial interpolants in Section 2.2.2. In some cases we cannot provide rigorous arguments but present observations obtained from numerical experiments. The main goal of this chapter is to point out to some of the most frequent problems when approximating with computed rational interpolants, and we enumerate them in Section 5.2. The main contribution of this chapter is a novel characterisation of the difference between the exact behaviour of rational interpolants and their behaviour in floating-point arithmetic (see item 4 in that list). In particular, we relate the information of the singular values of the Z matrix (cf. Theorem 4.1) with the validity of the computed rational approximant (see Figure 5.9 below). To round out the exposition of the analysis of rational approximation, we present, in Section 5.3, the Padé and Chebyshev–Padé approximants, which are obtained from the Taylor and Chebyshev coefficients of the target function, respectively.

5.1 Convergence analysis of rational interpolation

The arguments used to explain the convergence to analytic functions of polynomial interpolants can be used again with little modification to derive the convergence to meromorphic functions of rational interpolants with fixed denominator degree. The following theorem, although presented differently, is essentially due to Saff in 1972 [144, 145].

Theorem 5.2. *Let f be a meromorphic function in $\overline{E}_\rho$, the closed region bounded by a Bernstein ellipse with parameter $\rho > 1$, with precisely ν simple and distinct poles $\{b_j\}_{j=0}^{\nu}$. Let $\{r_m\}$ be a sequence of rational interpolants of type $[m/\nu]$, associated with a triangular sequence $\mathbf{X}$ with limiting Chebyshev distribution* (2.30). *Then $\{r_m\}$ converges to f uniformly, and more precisely,*

$$\|f - r_m\| = \mathcal{O}(\rho^{-m}).$$

Proof. Define the polynomial $g(z) = \prod_{j=1}^{\nu}(z - b_j)$ and let $p_m \in \mathcal{P}(m)$ and $q_m \in \mathcal{P}(\nu)$ be the numerator and denominator polynomials of r_m that satisfy the linear conditions (4.2), i.e.

$$p_m(x_j) = f(x_j)q_m(x_j), \quad \text{for } j = 0, \dots, N, \tag{5.1}$$

where $[x_0, \dots, x_{N+1}]$ is the Nth row of $\mathbf{X}$ and $N = m+\nu$. We assume that q_m is normalised in such a way that the the coefficient of largest modulus is 1. Consider the function $h_m(z) = g(z)f(z)q_m(z)$, analytic in $\overline{E}_\rho$ (notice how the polynomial g is used to annihilate the poles of f). From (5.1) it follows that $h_m(x_j) = g(x_j)p_m(x_j)$, $j = 0, \dots, N$, i.e. the polynomial $gp_m \in \mathcal{P}(N)$ interpolates h_m at $N + 1$ points.

Since h_m is analytic in $\overline{E}_\rho$, we use once more the Hermite integral formula, from which we obtain

$$h_m(x) - g(x)p_m(x) = \frac{\ell(x)}{2\pi i}\int_{E_\rho} \frac{h_m(t)}{\ell(t)(t-x)}dt, \quad x \in [-1, 1]. \tag{5.2}$$

From the same argument as in Theorem 2.5 we have

$$|h_m(x) - g(x)p_m(x)| \le K \max_{t\in E_\rho}\left|\frac{\ell(x)}{\ell(t)}\right|, \quad x \in [-1, 1], \ t \in E_\rho,$$

and since $\mathbf{X}$ has a limiting Chebyshev distribution it follows from (2.33) that

$$|h_m(x) - g(x)p_m(x)| \to 0 \ \text{ as } m \to \infty, \tag{5.3}$$

and in particular $\|h_m - gp_m\| = \mathcal{O}(\rho^{-N})$. Thus, the theorem is established if we prove that $|q_m(z)g(z)|$ is uniformly bounded from below on $[-1, 1]$.

From the normalisation of each q_m it is evident that $\{q_m\}$ is a sequence in the closed ball of $\mathcal{P}(\nu)$ of polynomials with coefficients of modulus less or equal than 1. It follows then that $\{q_m\}$ has a subsequence that converges uniformly to a limiting function, necessarily a polynomial of degree less or equal than ν. Let q_∞ be the limiting polynomial of such subsequence $\{q_{m_i}\}$, $\{m_i\}$ an ascending sequence of positive integers. Hence, from (5.2) we have

$$|(gfq_\infty - gp_{m_i})(x)| \le |(gfq_\infty - h_{m_i})(x)| + |(h_{m_i} - gp_{m_i})(x)| \to 0, \ \text{ as } m_i \to \infty.$$

Since $g(b_i)f(b_i) \neq 0$ but $g(b_i)p_{m_i}(b_i) = 0$, it follows necessarily that the zeros of q_∞ coincide with the zeros of g. As q_∞ is an arbitrary limiting polynomial, the zeros of $\{q_m\}$ tend to the zeros of g in $\overline{E}_\rho$ and $|q_m g|$ is bounded below on any compact subset of $\overline{E}_\rho \setminus \{b_j\}$, particularly on $[-1, 1]$. □

Let us illustrate this theorem with a numerical experiment. Consider the function

$$f(x) = \frac{\sqrt{x+3}}{p_1(x)p_2(x)p_3(x)p_4(x)}, \tag{5.4}$$

where p_1, p_2, p_3, and p_4 are polynomials defined by

$$p_1(x) = (x - 0.1)^2 + 0.05, \quad p_2(x) = (x + 0.5)^2 + 0.3,$$
$$p_3(x) = (x - 0.7)^2 + 1.9, \quad p_4(x) = x^2 + 4.$$

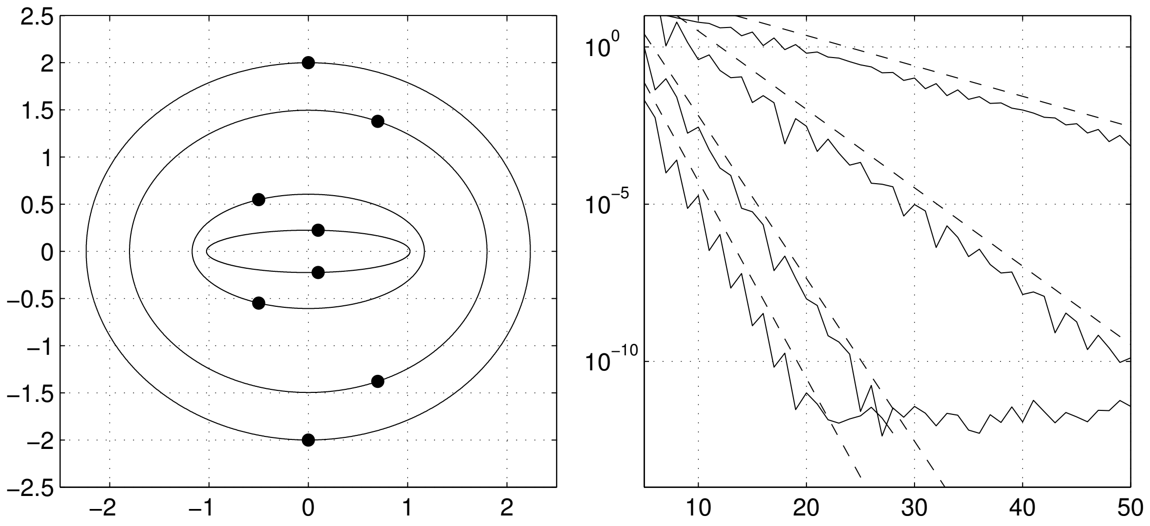

Figure 5.2: Uniform convergence of rational interpolants in Chebyshev points to the meromorphic function f in (5.4). Left: Simple poles of f and corresponding Bernstein ellipses with parameters $\rho_1, \ldots, \rho_4$ (from inner to outer). Right: Error for interpolants of type $[m/0]$, $[m/2]$, $[m/4]$, and $[m/6]$ (from top to bottom). The dashed lines indicate the rates $\mathcal{O}(\rho_i^{-m})$, $i = 1, \ldots, 4$.

In Figure 5.2 we plot the four pairs of poles introduced by the polynomials defining the denominator of f and the corresponding Bernstein ellipses that cross each pair. We also illustrate the error in the interpolants of type $[m/0]$, $[m/2]$, $[m/4]$, and $[m/6]$ for increasing m on grids of Chebyshev points. For the first case, $[m/0]$, Theorem 2.5 asserts that the rate of convergence is $\mathcal{O}(\rho_1^{-m})$, where ρ_1 is the parameter of the Bernstein ellipse where the function is free of poles. In this case, the zeros of p_1 determine $\rho_1 \approx 1.24960$, the inner ellipse. Theorem 5.2 explains the rest of the figure. The function f has 2, 4, and 6 poles inside the regions bounded by Bernstein ellipses with parameters $\rho_2 \approx 1.77516$, $\rho_3 \approx 3.29587$, and $\rho_4 \approx 4.23606$ respectively. The rates of convergence for the rational interpolants on the second, fourth, and sixth row of the interpolation table are $\mathcal{O}(\rho_2^{-m})$, $\mathcal{O}(\rho_3^{-m})$, and $\mathcal{O}(\rho_4^{-m})$ respectively.

For approximation on the unit disk, Theorem 5.2 is easily modified to consider grids of roots of unity by noticing that the logarithmic potential of their limiting distribution is constant in complex disks [144].

Theorem 5.3. *Let f be a meromorphic function in the closed disk $\overline{\mathcal{S}}_\rho$ of radius $\rho > 1$, with precisely ν simple and distinct poles $\{b_j\}_{j=0}^{\nu}$. Furthermore, let $\{r_m\}$ be a sequence of rational interpolants in roots of unity of type $[m/\nu]$ with ν fixed. Then $\{r_m\}$ converges to f uniformly on the unit disk $\overline{\mathcal{S}}$, and more precisely*

$$\|f - r_m\|_{\bar{\mathcal{S}}} = \mathcal{O}(\rho^{-m}).$$

In Figure 5.3 we plot the interpolating error in roots of unity for the function

$$f(z) = 100\frac{\log(3-x)}{(1.5^4 - x^4)(2.8^5 - x^5)}. \tag{5.5}$$

Two groups of error curves with different rates of decay are distinguishable. The upper group consists of the error for interpolants of type $[m/\nu]$, $\nu = 0, \ldots, 3$ and m between 1 and 50. In these four cases the decay of the error has the same rate $\mathcal{O}(\rho_1^{-m})$, where $\rho_1 = 1.5$ is the radius of the nearest circle with singularities to the unit disk. To overcome the four simple poles it is necessary to increase the degree of the denominator. The lower group consists of the interpolants of type $[m/\nu]$, $\nu = 4, \ldots, 8$ and m between 1 and 50. In this case the rate of convergence is improved to $\mathcal{O}(\rho_2^{-m})$, where $\rho_2 = 2.8$ is the radius of the external circle.

In the case where the function is meromorphic with poles inside the unit disk, an analogous result on convergence on the rows of the rational interpolation table can be established. Saff and Walsh [147] established that if the target function is meromorphic with exactly ν poles located inside the unit disk and continuous on the unit circle, the rational interpolant in roots of unity of type $[m/\nu]$ converges uniformly. According to [147], this result generalises a theorem by Fejér who proved that polynomial interpolants in roots of unity converge for functions analytic in the interior of the unit disk and continuous on the unit circle.

Notice the significance of this capability, which we illustrate in Figure 5.3: whilst for functions with poles inside the unit disk any polynomial approximant diverges, the rational interpolants of the appropriate type converge geometrically.

Rational interpolants converge uniformly in regions off the function's domain, in the same way as polynomials (cf. Theorem 2.10) but, once more, with the advantage of bypassing poles. Theorems 5.2 and 5.3 have the following straightforward generalisation.

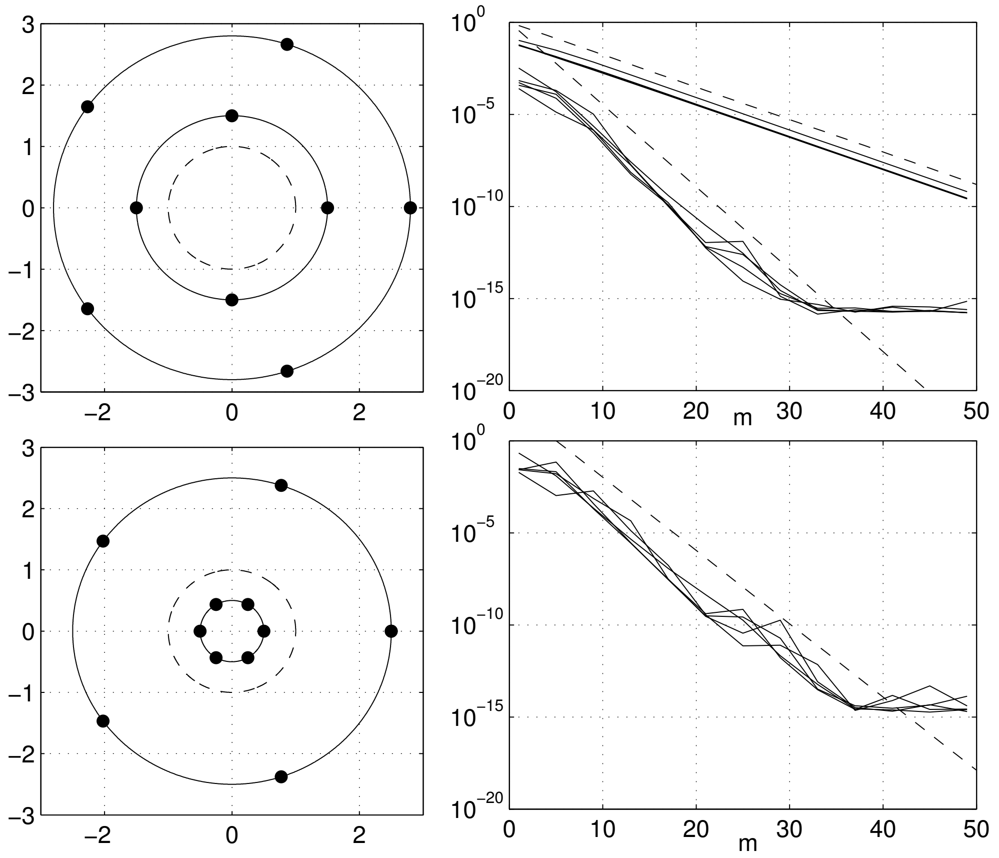

Figure 5.3: Upper panels: Uniform convergence of rational interpolants in roots of unity to the meromorphic function f in (5.5). Left: Simple poles of f and corresponding circles of radius ρ_1 and ρ_2. Right: Error for the interpolants of type $[m/\nu]$, $\nu = 0, \ldots, 3$ (upper group) and $\nu = 4, \ldots, 8$ (lower group). The dashed lines indicate the rates $\mathcal{O}(\rho_i^{-m})$, $i = 1, 2$. Lower panels: Uniform convergence of rational interpolants in roots of unity to a similar meromorphic function as f in (5.5) but with the denominator modified to have six simple poles on the circle of radius $\rho_1 = 0.5$ and five simple poles on the circle with radius $\rho_2 = 2.5$. Left: as above. Right: Error for the interpolants of type $[m/\nu]$, $\nu = 6, \ldots, 10$. The dashed line indicates the rate $\mathcal{O}(\rho_2^{-m})$ due to the poles in the outer circle. The rational interpolants with lower denominator degrees diverge.

Theorem 5.4. *If the hypotheses of Theorem* 5.2 *are satisfied, then* $\{r_m\}$ *converges to* f *uniformly on any compact subset of* $\overline{E}_\rho \setminus \{b_j\}_{j=0}^{\nu}$. *Moreover, if* $E_{\rho'}$ *is a* ρ'*-ellipse with* $1 \le \rho' < \rho$ *and such that* $E_{\rho'} \cap b_j = \emptyset$, $j = 1, \ldots \nu$, *then*

$$\|f - r_m\|_{E'_\rho} = \mathcal{O}\big((\rho'/\rho)^m\big).$$

Analogously, if the hypotheses of Theorem 5.3 *are satisfied, then* $\{r_m\}$ *converges to* f *uniformly on any compact subset of* $\overline{\mathcal{S}}_\rho \setminus \{b_j\}_{j=0}^{\nu}$. *Moreover, if* $\mathcal{S}_{\rho'}$ *is a circle with radius* ρ', $1 \le \rho' < \rho$ *and such that* $\mathcal{S}_{\rho'} \cap b_j = \emptyset$, $j = 1, \ldots, \nu$, *then*

$$\|f - r_m\|_{\mathcal{S}'_\rho} = \mathcal{O}\big((\rho'/\rho)^m\big).$$

Proof. The first part of the theorem follows the same argument in the proof of Theorem 2.10 together with the fact that the expression $|q_m g|$ in the proof of Theorem 5.2 is bounded below on any compact subset of $\overline{E}_\rho \setminus \{b_j\}_{j=0}^{\nu}$. The second part of the theorem uses the same arguments. $\square$

Clearly, rational interpolants provide a suitable method for evaluating the analytic continuation of a function, and Theorem 5.4 gives us precise estimates of its accuracy. Figure 5.4 shows the error of the rational interpolant in Chebyshev points with fixed denominator degree $\nu = 4$ when evaluated in the complex plane. As we move towards the ρ_3-ellipse that intersects the zeros of p_3, the ratio (ρ'/ρ_3) approaches 1, slowing its convergence on the ρ'-ellipse. See Figure 5.5 for a similar example with an interpolant on roots of unity.

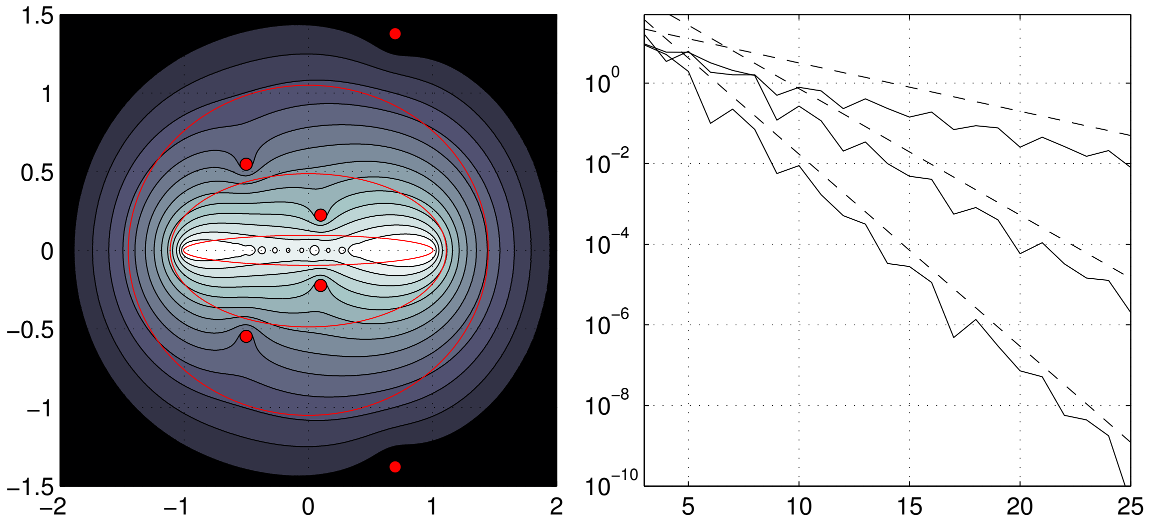

Figure 5.4: Accuracy of analytic continuation through rational interpolation in Chebyshev points for the function f of (5.4). Left: Contour plots for the error with an interpolant of type [25/4]. The contours correspond to errors of approximate size $10^0, 10^{-1}, \dots, 10^{-13}$ (from to darkest to lightest). The dots indicate the locations of the 6 singularities of f closest to the interval. Right: Error for the interpolants of type $[m/4]$, $m = 3, \dots, 25$ on the ρ'-ellipses with $\rho' = 1.1$ (lower curve), 1.6 (middle curve), and 2.5 upper curve. The ellipses are plotted with red lines on the left panel. The dashed lines on the right panel indicate the rates $\mathcal{O}\big((\rho\prime/\rho_3)^{-m}\big)$.

The proofs of Theorems 5.2–5.4 show that the ν zeros of the denominator of the rational interpolant approach the poles of the target function as the degree of the numerator increases. For Chebyshev points the rate of this approach is geometric with factor ρ'/ρ, where ρ' is the parameter of the Bernstein ellipse where the singularity is located and ρ is the parameter of a Bernstein ellipse that encloses the ν poles. This is a simple consequence of evaluating (5.2) in the poles $\{b_j\}$ and the usual bound on $|\ell(t)|$, $t \in E_\rho$. Exactly the same argument applies for approximation on the unit circle with interpolants in roots of unity. For an application of this capability, see for example the method of Kravanja, Sakurai, and

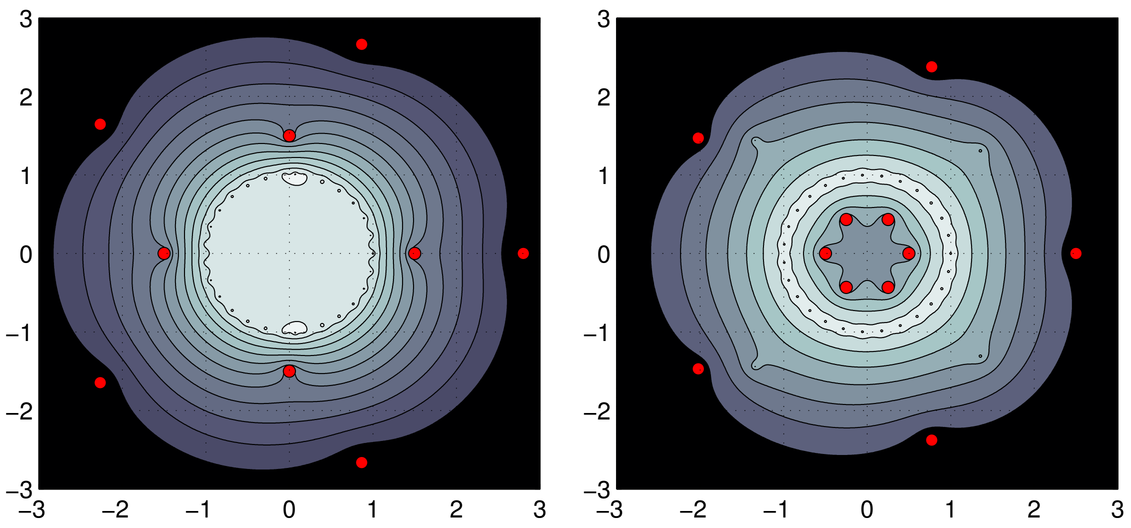

Figure 5.5: Contour plots for the error of analytic continuation through rational interpolation in roots of unity of type [20/4] (left) and [20/6] (right) for the top and bottom functions depicted in Figure 5.3 respectively. The dots indicate the locations of the poles in each case. The contours, from darkest to lightest, correspond to errors of size $10^{-1}, 10^{-2}, \ldots, 10^{-12}$ (left) and $10^{-1}, 10^{-2}, \ldots, 10^{-7}$ (right). In the right panel, the ring-shaped area surrounding the unit circle is resolved with approximate accuracy of 10^{-7}.

van Barel [84], which uses rational interpolants in roots of unity to locate the zeros of a meromorphic function inside of a Jordan curve.

The issues discussed so far concern the derivation of the rate of convergence from the analytic structure of the function. Inverse results for rational interpolants, in which the rate of convergence determines the analytic structure, are more difficult to obtain. An inverse result for rows of rational interpolants was given by Karlsson and Saff [82], and it requires not only a particular form of the limiting distribution of the triangular sequence of points but also an additional condition on its *generalised Taylor series.* Further results were obtained by Martínez Finkelshtein [102, 103] and Le Ba Kkhan' Chin' [88].

There seem to be fewer results on the convergence analysis of rational interpolation for non-analytic functions. Nevertheless, the particular study of $|x|$ has shown that the behaviour of rational functions can differ significantly from that of polynomials. While Theorem 2.8 shows that polynomial interpolants in Chebyshev points to $|x|$ converge at the rate $\mathcal{O}(1/N)$, Brutman and Passow [24] proved that rational interpolants on the diagonal converge at the slightly faster rate $\mathcal{O}(1/(N \log N))$. More striking is the case of equidistant nodes: Unlike the polynomial case, for which the interpolant diverges at every point except -1, 0 and 1, Werner [181] showed that the rational interpolant on the diagonal converges at the rate $\mathcal{O}(1/(N \log N))$.

The original observation that rational approximants can greatly improve convergence over polynomials was made by Newman [120], who showed that the error to $|x|$ with certain rational approximants of type $[n/n]$ was bounded above by $3e^{-\sqrt{n}}$, $n \geq 5$. Newman approximants can in fact be understood as rational interpolants in the grid

$$\boldsymbol{\eta} = [-1, -\eta, \ldots, -\eta^{n-1}, 0, \eta^{n-1}, \ldots, \eta, 1]^{\mathsf{T}}, \quad \text{with } \eta = e^{-1/\sqrt{n}}.$$

Xie and Zhou [183] proved that the the exact rate of approximation of these rational interpolants is $\mathcal{O}(n^{-1/2}e^{-\sqrt{n}})$. Note that the points of $\boldsymbol{\eta}$ cluster around the origin, approaching it at an exponential rate as n increases. Relying on this observation, Brutman [23] showed that a grid consisting of two sets of n transplanted Chebyshev points each, one in $(-1, 0)$ and the other in $(0, 1)$, would also cluster around the origin and its associated rational interpolant converges at a rate $\mathcal{O}(n^{-2})$.

Using the rational interpolation method presented in Chapter 4 we illustrate in Figure 5.6 the rates of convergence for rational interpolants to $|x|$ on Chebyshev, Newman, and adjusted Chebyshev points. The curves for the last two are very close to one another and their separation is only noticeable for very high degrees. Figure 5.6 also shows a conspicuous effect of rational interpolation: the smooth decay of the curves is suddenly truncated by numerical difficulties in each case (at $n = 15$, $n = 11$ and $n = 20$ for Chebyshev, Newman, and adjusted Chebyshev points respectively). In Section 5.2 we give an explanation of this disturbing anomaly caused by the computation in floating-point.

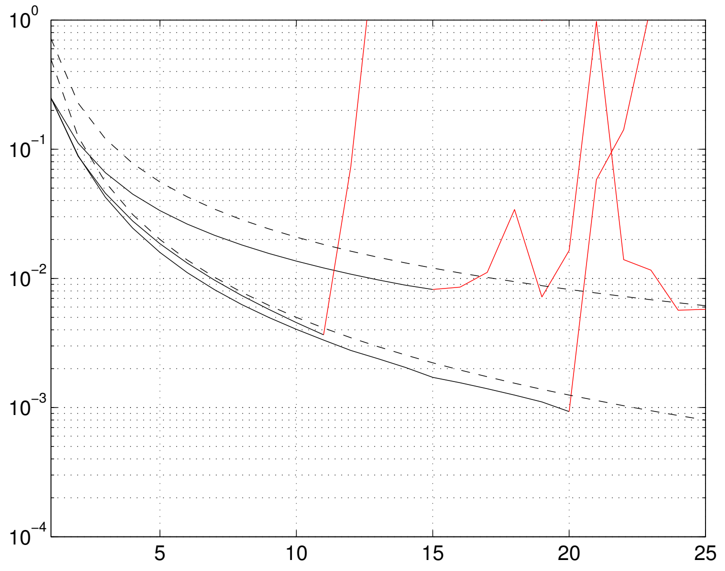

Figure 5.6: Rational interpolation error to $|x|$ on Chebyshev points (solid upper), Newman points (solid middle), and adjusted Chebyshev points (solid lower). The curves decay smoothly up to a certain point at which they start to diverge for reasons related to rounding errors. We plot in red the part of the curve that is not in agreement with the theoretical rates of convergence. The dashed lines indicate the rates $\mathcal{O}(n^{-1})$ and $\mathcal{O}(n^{-2})$.

The last remark in this section concerns the behaviour of row rational interpolants to discontinuous functions. The overshoot of the Gibbs phenomenon for sign(x) is in this case smaller than in the polynomial case and the amplitude of the wiggles is reduced, see Figure 5.7. From numerical experiments, we have observed that the magnitude of the overshoot for interpolants to sign(x) in Chebyshev points of types $[m/2]$ and $[m/4]$, approximately converges to $1.032\dots$ and $1.014\dots$. The other oscillations are also noticeably smaller. The following table shows the magnitude of the five largest local maxima for the interpolants of types [81/0], [81/2] and [81/4].

[81/0]	[81/2]	[81/4]
1.065745964988479	1.032133536823809	1.014581379561677
1.012685442260004	1.000268831918086	1.000013953744225
1.005172073627946	1.000023662082417	1.000000249083761
1.002773080323939	1.000004628457015	1.000000015101623
1.001714996200490	1.000001360853672	1.000000001794780

None of these observations seem to have been made before for the Gibbs phenomenon in rational interpolants in Chebyshev points or any other grid.

We have observed that the degree of the numerator cannot be increased freely without reaching a moment when the error looses this behaviour and very large overshoots start to appear. This effect is even stronger with interpolants whose denominator degrees are larger than 4. In the next section we explain the cause of this phenomenon, which is the same that produces the discrepancy of the computed error in Figure 5.6.

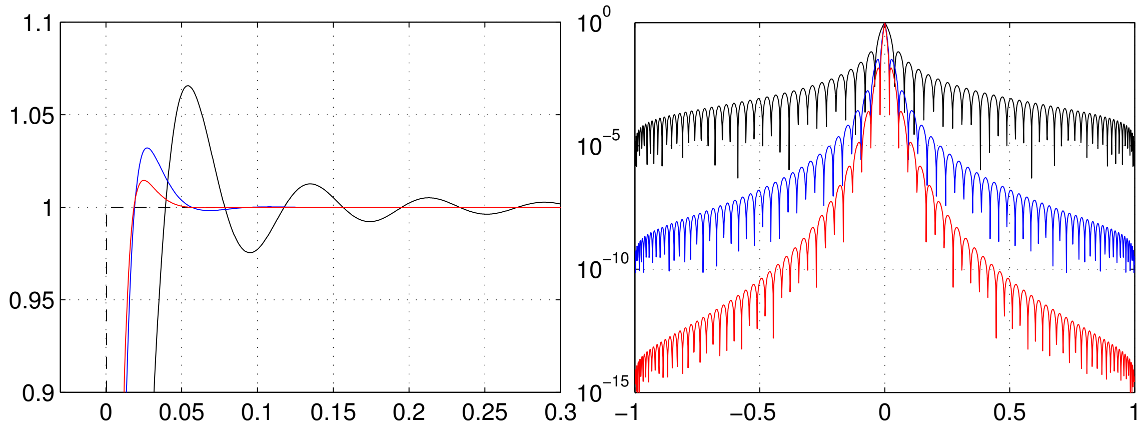

Figure 5.7: Gibbs phenomenon for interpolants in Chebyshev points of types [80/0] (black), [80/2] (blue) and [80/4] (red) to sign(x). Left: Detail of the overshoot. Right: Global error.

5.2 Practical considerations in approximation by rational interpolants

Numerical experimentation hints towards rational interpolation being a powerful but fragile tool for approximation. In many cases it reproduces the good properties mentioned thus far, but more often than not it also fails due to a confluence of various unrelated issues. A first step to establishing its limitations on a practical level is to relate the original nonlinear problem (4.1) and the solution of its linearisation (4.2), for example by computing the kernel of the matrix Z in Theorem 4.1. The following theorem is due to Maehly and Witzgall [99] and can also be found, for example, in [182] and [70].

Theorem 5.5. *Let $p \in \mathcal{P}(m)$ and $q \in \mathcal{P}(n)$ satisfy the linear equations* (4.2). *Then p and q are of the form $p(x) = (\overline{p}sv)(x)$, $q(x) = (\overline{q}sv)(x)$, where*

- *$\overline{p}$ and $\overline{q}$ are relatively prime polynomials, unique up to normalisation, and of exact degrees $\overline{m}$ and $\overline{n}$, $0 \leq \overline{m} \leq m$ and $0 \leq \overline{n} \leq n$*
- *s is a monic polynomial of degree δ_s, where $0 \leq \delta_s \leq \delta = \min(m - \overline{m}, n - \overline{n})$, whose zeros are the interpolation points x_j such that $q(x_j) = 0$*
- *v is an arbitrary polynomial of degree $\delta_v = \delta - \delta_s$*

If Z is the $n \times (n+1)$ matrix of Theorem 4.1, then

$$\operatorname{rank}(Z) = n - \delta + \delta_s,$$

which is equivalent to saying that the number of nonzero singular values of Z is equal to the number of common factors of p/q.

Proof. The first part of the theorem follows immediately from the definitions of $\overline{p}$, $\overline{q}$ and s. The formula for the rank of Z follows from the observation that v is arbitrary, hence the nullity of Z is $\delta_v + 1 = \delta - \delta s + 1$. (The value δ is known as the *defect* and the rational function p/q is *nondegenerate* if $\delta = 0$.) □

In this section we present four issues that arise in practical approximation by rational interpolants.

1. Unattainable points. If none of the values $q(x_j)$ is zero, then we can define the rational function $r = p/q$ that interpolates the data $\mathbf{f}$ in the grid $\mathbf{x}$. However, if $q(x_j) = 0$ for some j, i.e. x_j is a zero of s, then $p(x_j) = 0$ and both p and q have the common factor $(x - x_j)$ that can be cancelled out. Thus, in this case the rational function r in its reduced form may or may not satisfy the interpolation condition, i.e. $\overline{p}(x_j)/\overline{q}(x_j) = f_j$ may not hold for a certain x_j. Grid points where the interpolation conditions (4.1) cannot be satisfied are called *unattainable points* and their presence indicates the nonexistence of a solution to Cauchy's interpolation problem.

Notice that $q(x_j) = 0$ is a necessary but not sufficient condition for a point x_j to be unattainable. Unattainable points can be detected *a posteriori* simply by evaluating the rational function at interpolation points where q is zero.

Many examples of rational interpolants with unattainable points can be tracked to blocks of more than one cell in the interpolation table where the rational interpolant is the same[2]. For example, for symmetric grids, even and odd functions determine tables with blocks of size 2×2. The patterns arising in the rational interpolation table have been studied by Wuytack [182].

2. Poles of the interpolant. The zeros of q may appear anywhere in the complex plane. In particular, when interpolating values defined on $[-1, 1]$, it may happen that the zeros of q are located inside the interval. See Figure 5.8.

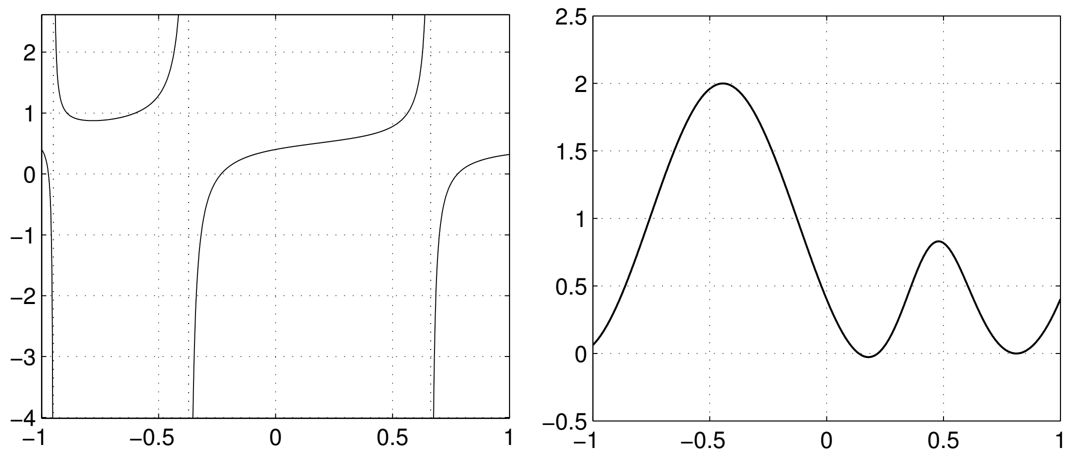

Figure 5.8: Rational interpolation of $f(x) = 1 - \sin(5|x - 0.5|)$ in Chebyshev points of the first kind. Left: Interpolant in $\mathcal{R}(3, 3)$. Notice that q has zeros at approximately -0.949409857044933, -0.371655244598090, and 0.663444249729421. Right: Interpolant in $\mathcal{R}(6, 6)$. In this case q does not have any zeros in the interval, and the error is approximately 0.182430032146706.

3. Ill-conditioned grids. The evaluation of the interpolant with the rational barycentric formula (4.9) usually requires the computation of the polynomial barycentric weights[3]. These weights are sensitive to the distribution of the nodes and for certain grids, for example a set of equispaced points, their values vary exponentially. This may results in weights that are numerically negligible thus affecting the representation of the interpolant.

4. Artificial common factors. A frequent source of difficulties in rational interpolation

[2]If instead of constructing the table from the rational functions that satisfy the linear condition we use the nonlinear condition, certain cells might not be defined due to unattainable points. As observed by Gutcknecht [71, 72, 73], the structure of these tables is certainly more complex, with blocks of identical functions not necessarily arranged in square shapes.

[3]A notable exception is the method by Berrut and Mittleman [13], which circumvents this calculation.

is the polynomial v which collects the common factors of p and q other than those due to s. From Theorem 5.5 it follows that δ_v can be detected by looking at the rank (or the nullity) of Z. In practice, however, it may happen that Z is numerically rank-deficient while there are nevertheless no common factors in p and q. In infinite precision Z would be of full rank, but linear independence is lost due to finite precision effects.

Consider for example the function [13]

$$f = \frac{\exp((x+1.2)^{-1})}{1+25x^2}. \tag{5.6}$$

Its interpolant of type [18/18] in 37 Chebyshev points of the first kind in $[-1, 1]$ does not have common factors, as can be verified by symbolic computation software. In the top left panel of Figure 5.9 we plot 16 pairs of zeros of p and q computed numerically with Maple using 100 decimal digits of precision (the others are two zeros of p at infinity and two zeros of q at $\pm i/5$).

On the other hand, if we use ordinary IEEE double-precision arithmetic to compute the same rational interpolant, the Matlab command `rank` determines that the rank of Z is 9. Since $q(x_j) \neq 0$, $j = 0, \ldots, 37$, we should have that $\delta_s = 0$ and $\delta = \delta_v = 9$. In the top right panel of Figure 5.9 we plot 16 pairs of zeros of p and q computed in IEEE double-precision arithmetic (the others are two zeros of p at approximately $\pm 1.535 \times 10^5$ and two zeros of q at $\pm 0.2i$). Clearly the zeros and poles of the computed rational interpolant differ completely from the true ones. Notice that there are 9 pairs of common factors, in agreement with the computed rank of Z.

The lower panels of Figure 5.9 can give us more insight into this phenomenon. We compute the interpolants of f in $\mathcal{R}(k, k)$, $k = 1, \ldots, 18$, that is, the first 18 elements in the diagonal of the rational interpolation table. For each k, we display all the singular values of the associated matrix Z, normalised with respect to the largest one, by joining them with a line (as a reference, all the singular values when $k = 18$ are marked with a white circle, and for $k = 1, \ldots, 17$, only the smallest one is marked with a black dot).

Similarly as before, the left panel shows the singular values obtained with Maple using 100 decimal digits, and the right panel shows the singular values computed in double precision. A striking feature of these plots is the fast decay of the singular values as we increase k. When $k = 18$, the smallest singular value is approximately 1.4105×10^{-60}. The right panel shows that when $k \geq 9$ the magnitude of the smallest singular values is in the order of machine precision $\epsilon_M \approx 2.22 \times 10^{-16}$. In particular, when $k = 18$, only the first 9 are larger than ϵ_M, which corresponds to the numerically computed rank of Z. Moreover, the remaining values introduce noise into the polynomials p and q, and since the singular values of Z are numerically negligible, they correspond to common terms in the polynomial v.

We attribute the behaviour of the error in Figure 5.6 to the appearance of artificial common factors. In each case, the error of the computed interpolant deviates from the

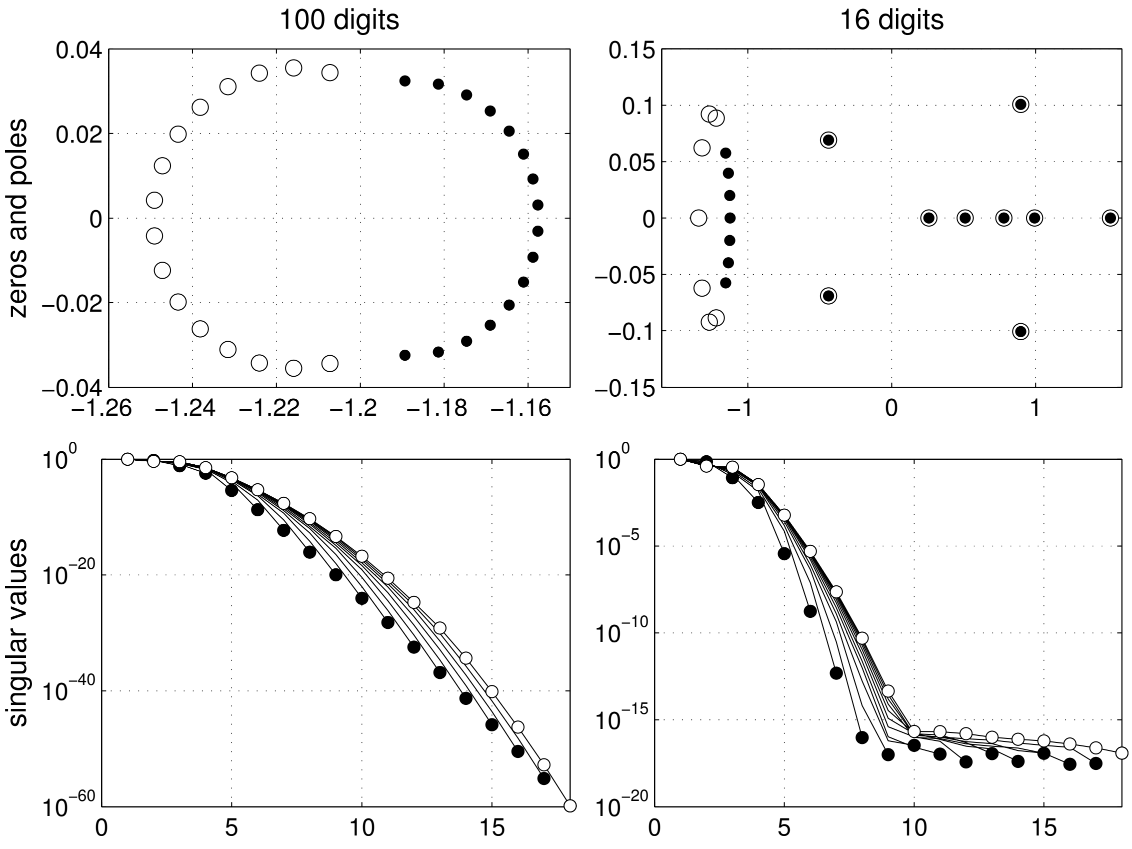

Figure 5.9: Rational interpolation of (5.6) in Chebyshev points of the first kind. Left panels show computations using Maple with 100 decimal digits and right panels show computations in ordinary IEEE double-precision arithmetic. Upper: Sixteen pairs of zeros of the numerator (larger white circles) and denominator (smaller black dots) of the [18/18] interpolant. Two other pairs are outside the plotted region. Lower: Singular values of Z, normalised with respect the largest one, for the $[k/k]$ interpolant $k = 1, \ldots, 18$. For $k = 18$ the values are marked with a white circle, and for the rest only the smallest ones are marked with black dots. The nine pairs of overlapping zeros of p and q in the upper right panel arise due to the nine singular values below machine precision in the lower right panel.

theoretical bound precisely when the smallest singular value reaches the order of machine precision. A similar effect happens for the Gibbs phenomenon with rational interpolants with a very large numerator or with a degree of the denominator larger than 4. For the function (5.6), the error keep decreasing and it is not obvious from Figure 5.10 that the computed rational interpolant in Chebyshev points of the first kind has artificial common factors, except perhaps for a kink at $k = 8$, when the smallest singular value reaches machine precision (see the lower right panel of Figure 5.9). On the other hand, no effect is noticeable for rational interpolants in Chebyshev points of the second kind to the same function.

The discrepancy between computed and theoretical rational interpolants can be magnified by the wrong choice of the basis used to represent p and q. Figure 5.11 shows similar

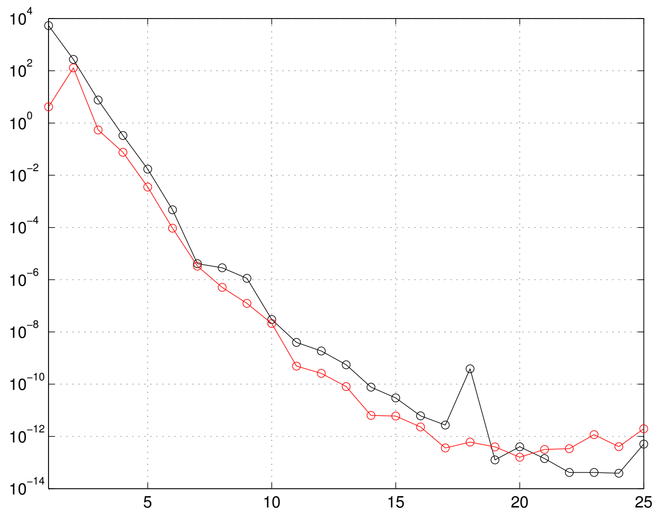

Figure 5.10: Error of rational interpolation of type $[k/k]$, $k = 1, \ldots, 25$ to (5.6) in Chebyshev points of the first (black) and second kind (red). For points of the first kind, a small kink appears when $k = 8$, which is precisely when the smallest singular value reaches machine precision on the lower-right panel of Figure 5.9. For points of the second kind the smallest singular value also reaches machine precision when $k = 8$, however nothing in the error seems suspicious.

plots to the one in the right panel of Figure 5.9, with the singular values of the Z matrix (4.21) for the rational interpolation of type $[2k/2k]$, $k = 1, \ldots, 25$, in Chebyshev points of the second kind for the function (5.6). For the plot in the left panel we use the monomial basis, and in the right panel we use the Chebyshev basis, i.e. the modified method of Section 4.2.1. Notice how for monomials, the tail of the singular values grows as we increase k. For the Chebyshev basis, on the other hand, the first singular values seem to converge while the tail remains close to machine precision as we increase k.

This phenomenon typically appears when either degree of the numerator or denominator increases, and is most noticeable when computing elements from the diagonal of the table: the singular values decrease very quickly, reaching eventually machine precision, which in turn introduces artificial common terms in p and q that make the zeros and poles of the rational interpolant deviate from the expected ones. It seems that these observations have not been discussed before in the literature.

When the artificial common factors are introduced in the interpolant we can attempt to combine the (numerically) linearly independent vectors in the kernel of Z to produce a reduced denominator q, i.e. a coefficient vector $\boldsymbol{\beta}$ whose last entries are zero. This modifies the problem of computing an interpolant of type $[m/n]$ into one of type $[m/\nu]$, where $\nu < n$ is the largest degree of the denominator such that no artificial common factors are present.

For example, let σ_k, $k = 0, \ldots, n-1$, be the singular values of Z and V the matrix of the corresponding right singular vectors. If $\sigma_k < \epsilon_M$ for $k = \nu, \ldots n-1$, the columns of

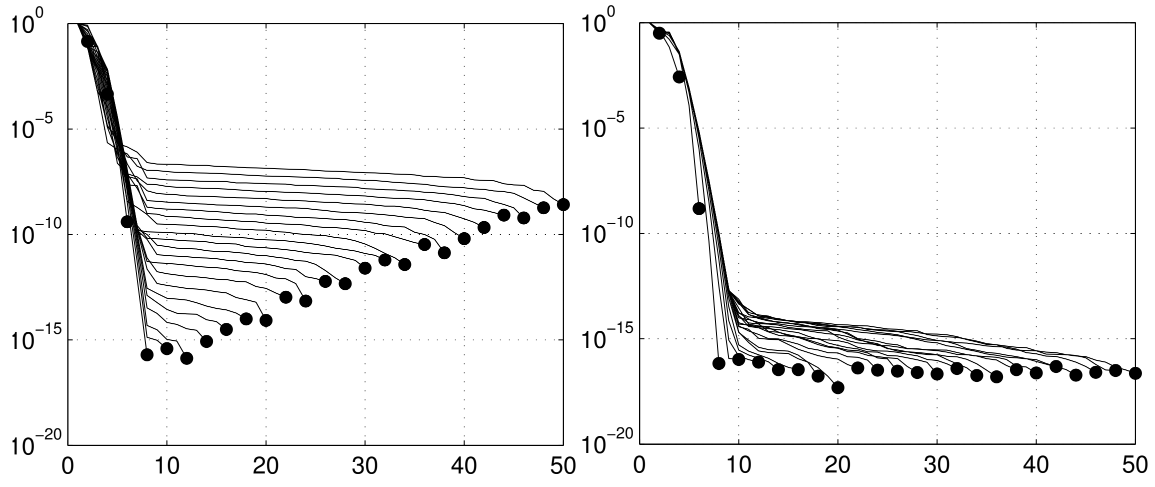

Figure 5.11: Normalised singular values of the Z matrix in (4.21) for the interpolants of (5.6) in $\mathbf{y}^{(2)}$ of type $[2k/2k]$, $k = 1, \ldots, 25$. For each k, the smallest singular value is marked with a black dot. Left: Monomials basis. Right: Chebyshev basis (this is equivalent to the modified method for $\mathbf{y}^{(2)}$ presented in Section 4.2.1). When using the original method, i.e. with the polynomials of Eisinberg and Fedele, the plot obtained is indistinguishable from the one in the right panel.

$V_{\nu:n}$ correspond to a basis for the kernel of Z. Let R be the triangular matrix of the QR factorisation of $(PV_{\nu:n})^{\mathsf{T}}$, where P is the anti-diagonal matrix that flips $V_{\nu:n}$ upside-down. Then, the row spaces of $(PV_{\nu:n})^{\mathsf{T}}$ and R are the same, but the first $n - \nu$ entries of the last row of R are zero. Hence, we can use the last row of R as a vector of coefficients for a reduced q. If the null space `V` of Z has already been computed, the following two lines compute the vector of coefficients $\boldsymbol{\beta}$ of the reduced q:

```
>> [Q,R] = qr( flipud( V ).' );
>> beta = flipud( R(end,:).' );
```

We use this strategy to obtain reduced denominators of rational interpolants of type $[k/k]$, $k = 1, \ldots, 250$, in Chebyshev points of the first kind, for the function

$$f(x) = \frac{xg(x)}{\sinh(g(x))}, \quad x \in [-1, 1], \tag{5.7}$$

where

$$g(x) = \frac{\pi}{\omega}(x^2 - c^2), \quad \text{with } c = 0.6, \text{ and } \omega = 0.02.$$

In the left panel of Figure 5.12 we plot the error of the rational interpolant as k increases. The thin line shows the error of the rational function with a denominator of full degree. The bold line shows the error when modifying the code to reduce the degree of the denominator, which is presented in the right panel of the Figure. The degree of the denominator slowly increases reaching 40 when $k = 182$, after which it remains almost unchanged.

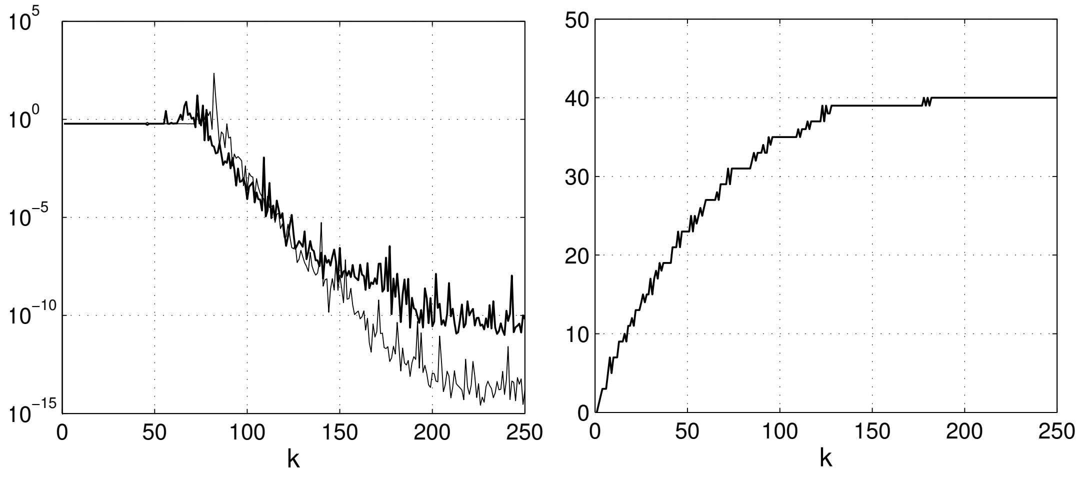

Figure 5.12: Rational interpolation of the function (5.7). Left: Error measured in a grid of 300 equispaced points using rational functions with full degree (thin line) and reduced degree (bold line). Right: Degree of the reduced denominator.

5.3 Padé and Chebyshev–Padé approximations

The coefficients of an infinite series representing the target function can be used to construct suitable approximations. This was the observation made in Section 2.3, where Taylor and Chebyshev series were truncated to obtain polynomial approximations. The rational approximants that are derived from infinite series are known as *Padé-type* or *maximum contact approximations*, and are obtained by enforcing the condition that the series representing the approximating and approximated functions coincide up to as many terms as possible. The name *Padé approximation* is reserved for the specific rational function derived from the Taylor series of an analytic function and thorough discussions regarding it can be found, for example, in [2, 20, 21, 64, 172]. More precisely, for a function f analytic at the origin, it is defined as the rational function $r = p/q \in \mathcal{R}(m,n)$, $p \in \mathcal{P}(m)$ and $q \in \mathcal{P}(n)$, such that [146]

$$f(x) - \frac{p(x)}{q(x)} = \mathcal{O}(x^{s+1}), \tag{5.8}$$

where s is as large as possible. This last requirement is commonly used in the definition of Padé approximants to avoid cases of non-uniqueness that may occur if we demand that $s = m+n+1$. As with rational interpolation, the Padé approximant is computed from the associated linear problem

$$q(x)f(x) - p(x) = \mathcal{O}(x^{m+n+1}). \tag{5.9}$$

Replacing f with its formal power series we obtain

$$(\beta_0 + \beta_1 x + \ldots + \beta_n x^n) \sum_{k=0}^{\infty} \gamma_k x^k = \alpha_0 + \alpha_1 x + \cdots + \alpha_m x^m + \mathcal{O}(x^{m+n+1}), \tag{5.10}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the coefficients of p and q respectively. Equating the coefficients of $x^{m+1}, \dots, x^{m+n}$ and setting $\gamma_k = 0$ if $k < 0$, we obtain the following linear system of n equations and $n+1$ unknowns:

$$\begin{array}{ccccccccc} \beta_m\gamma_{m-n+1} & + & \beta_{m-1}\gamma_{m-n+2} & + & \dots & + & \beta_0\gamma_{m+1} & = & 0, \\ \beta_m\gamma_{m-n+2} & + & \beta_{m-1}\gamma_{m-n+3} & + & \dots & + & \beta_0\gamma_{m+2} & = & 0, \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ \beta_m\gamma_m & + & \beta_{m-1}\gamma_{m+1} & + & \dots & + & \beta_0\gamma_{m+n} & = & 0. \end{array} \tag{5.11}$$

If we fix the value $\beta_0 = 1$, the coefficients $\beta_1, \dots, \beta_n$ correspond to the solution of the system

$$\begin{bmatrix} \gamma_{m-n+1} & \gamma_{m-n+2} & \cdots & \gamma_m \\ \gamma_{m-n+2} & \gamma_{m-n+3} & \cdots & \gamma_{m+1} \\ \vdots & \vdots & & \vdots \\ \gamma_m & \gamma_{m+1} & \cdots & \gamma_{m+n-1} \end{bmatrix} \begin{bmatrix} \beta_n \\ \beta_{n-1} \\ \vdots \\ \beta_1 \end{bmatrix} = - \begin{bmatrix} \gamma_{m+1} \\ \gamma_{m+2} \\ \vdots \\ \gamma_{m+n} \end{bmatrix}. \tag{5.12}$$

The solution is unique (up to a scalar) if the the matrix in (5.12) is nonsingular. The numerator coefficients $\boldsymbol{\alpha}$ follow from (5.10):

$$\begin{aligned} \alpha_0 &= \gamma_0, \\ \alpha_1 &= \gamma_1 + \beta_1\gamma_0, \\ \alpha_2 &= \gamma_2 + \beta_1\gamma_1 + \beta_2\gamma_0, \\ &\vdots \\ \alpha_m &= \gamma_m + \sum_{i=1}^{\min(m,n)} \beta_i\gamma_{m-i}. \end{aligned}$$

By Cramer's rule applied to system (5.11), it follows that the determinant of the matrix in (5.12) corresponds to the coefficient β_0. Thus, if this matrix is singular then $\beta_0 = 0$ and $q(0) = 0$, hence there is a factor x^l which will decrease the accuracy of the right-hand side of (5.9) to a smaller value than $m + n + 1$.

If `c` $= [\gamma_0, \dots, \gamma_{m+n+1}]$ is a vector with the first $m+n+1$ Taylor coefficients of f, the following six lines in Matlab compute the vectors `a` $= [\alpha_m, \alpha_{m-1}, \dots, \alpha_0]$ and `b` $= [\beta_n, \beta_{n-1}, \dots, \beta_1]$ with the coefficients for the polynomials p and q respectively, satisfying (5.9):

```
>> t = [zeros(1,n-m-1) c(max(1,m-n+2):m+n+1)]';
>> h = hankel(t(1:m),t(m:end-1));
>> b = [h\-t(m+1:2*m)]';
>> t = c(m+1:-1:1);
>> bt = conv(b,t(2:end));
>> a = t + [bt(max(m,1):end) 0];
```

Padé approximation has been the focus of much research, and its analytic theory has been developed much earlier than the theory of rational interpolation. For example, the analogue of Theorem 5.2 for the Padé case was established by de Montessus de Ballore in 1902 [37]—seventy years before Saff's proof. Since the condition of Padé approximants is the agreement of the rational function with the first terms of the Taylor series, this can be seen as an interpolation condition at the origin in the sense of Hermite. For this reason, the term *multipoint-Padé* approximation can be found in the literature to refer to rational interpolants, but for which coincident points in the grid are allowed.

Another related approximant, but perhaps less known, is the *Chebyshev–Padé approximation* (CP) defined as the rational function $r = p/q \in \mathcal{R}(m,n)$ which satisfies the nonlinear condition

$$f(x) - \frac{p(x)}{q(x)} = \mathcal{O}(T_s(x)), \tag{5.13}$$

where s is as large as possible. Here $\mathcal{O}(T_k(x))$ denotes a Chebyshev series with no term of degree less than k. Notice that is implicit in the definition that r has a Chebyshev expansion in $[-1,1]$ and therefore that it does not have poles in that interval. Unfortunately, unlike Padé approximation, CP approximants may not be unique.

The literature on CP is not extensive. The first written reference seems to have been made by Maehly in a couple of technical reports of the Institute of Advanced Study by the end of the 1950s, when he was working as a Principal Engineer for the von Neuman Electronic Computer Project [95, 96, 97]. However, these approximants, referred as CP-Maehly, do not satisfy the maximum contact condition but instead they solve a "cross-multiplied" system. More precisely, CP-Maehly approximants are rational functions $R \in \mathcal{R}(m,n)$ that satisfy a linearised version of condition (5.13). In particular, the numerator and denominator polynomials, P and Q respectively, satisfy

$$Q(x)f(x) - P(x) = \mathcal{O}(T_{m+n+1}(x)). \tag{5.14}$$

In standard Padé approximation, the rational function obtained from the linearised condition is equivalent to the one obtained from the nonlinear condition. Rational approximants defined from conditions (5.13) and (5.14), however, are not equivalent. If Q^{-1} exists as a Chebyshev series and we multiply (5.14) by Q^{-1}, the Chebyshev series on the right will have, in general, nonzero coefficients starting with $k = 0$ since the product rule of Chebyshev polynomials does not, in general, satisfy $T_i(x)T_j(x) = T_{i+j}(x)$ (cf. Section 2.1.2).

Figure 5.13 illustrates this point. We construct the chebfun of $f(x) = \sin(\exp(x)) + \exp(\sin(x))$ and compute the CP-Maehly approximant R of type [5/2] using the `chebpade` command with input argument `'maehly'`. In the left panel we plot the Chebyshev coefficients of $Qf - P$ and the Chebyshev coefficients of $f - P/Q$. The first 8 coefficients of the former are negligible, but not those of the later. This behaviour of the Chebyshev coefficients of $f - P/Q$, in which the first $m+n+1$ terms are small but nonetheless well above

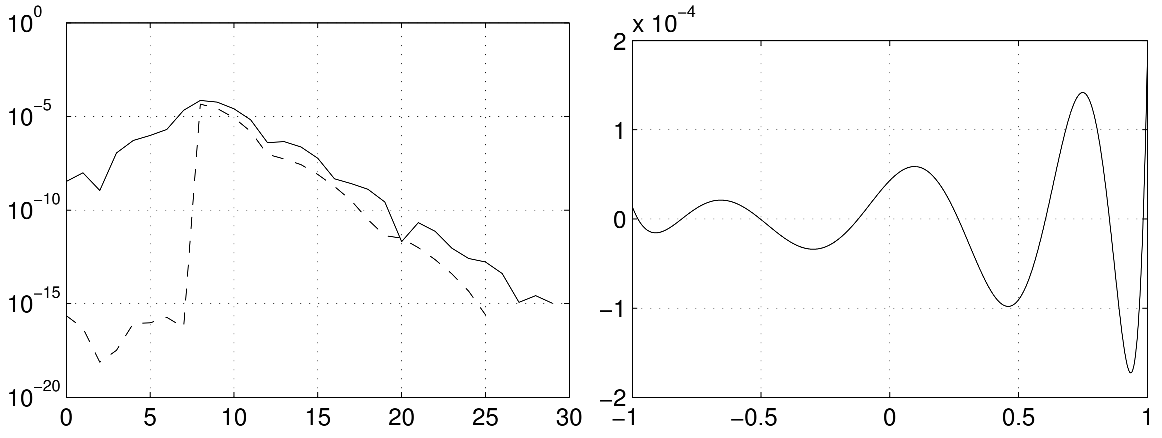

Figure 5.13: Left: Chebyshev coefficients of $(Qf - P)$ (dashed) and $f - P/Q$ (solid) where $f = \sin(\exp(x)) + \exp(\sin(x))$, and P/Q is the [5/2]-CP-Maehly approximant. Right: Error $f - P/Q$. The norm of the error is $1.829247647795462 \times 10^{-4}$.

machine precision, seems typical for CP-Maehly approximants. In the right panel we plot the error $f - P/Q$.

Regarding CP-Maehly approximants, Fleischer [47] tried to overcome some of its limitations by generating what Baker and Graves-Morris called a "*horrific system of nonlinear equations*" [2, p. 389]. Mason and Crompton [105] gave the coefficients of the rational function when the target function is expanded in second, third and fourth Chebyshev polynomials. Kaber and Maday [81] obtained analytical formulas for the approximants of the sign function when studying the Gibbs phenomenon in the approximation of discontinuous functions. Finally, Tee and Trefethen [162] used them to approximate the location of singularities in their rational spectral collocation method for solving differential equations.

The earliest reference on CP approximation that truly satisfies the Padé condition is a one-paragraph note by A. P. Frankel and William B. Gragg in 1973 [51], in which they briefly mention how it can be constructed by computing Padé approximations of Laurent series[4]. The following year, Clenshaw and Lord [28], following a different approach, published the first complete study on this type of CP approximation[5]. Gragg further developed his ideas on CP approximation and block structure of CP tables in [65], and his deduction of the CP conditions is the one most frequently used. The structure of the CP table was further studied in the 1980s by Geddes [58]. The development of the coefficients for CP approximants from expansions in other kinds of Chebyshev polynomials was given in [106]. The theory of convergence for CP-Maehly and CP approximation was studied by Suetin and general descriptions for both types of approximants are given in [2, Sec. 7.4].

[4]These Laurent-Padé approximations were more carefully defined and studied by Gragg and Johnson [66] and Bultheel [25]. These are the "natural" Padé approximants for functions analytic on an annulus.

[5]In the literature CP approximants are also referred to as CP-Clenshaw-Lord approximants.

An appropriate way of constructing CP approximants, as proposed by Geddes [58], is by first defining Padé approximations $\tilde{r}(w)$ of the corresponding Laurent series of f in the unit circle $\mathcal{S}$,

$$f(x) = \tilde{f}(w) = {\sum_{k=0}^{\infty}}' c_k w^k, \quad w \in \mathcal{S},$$

where x and w are related by the Joukowski transform $x = \frac{1}{2}(w + w^{-1})$. The approximants $\tilde{r}$ are functions in $\mathcal{R}(l, n)$ of the form

$$\tilde{r}(w) = \frac{\tilde{p}(w)}{\tilde{q}(w)} = \frac{\sum_{k=0}^{l} \tilde{\alpha}_k w^k}{\sum_{k=0}^{n} \tilde{\beta}_k w^k}, \tag{5.15}$$

where $l = \max\{m, n\}$, constructed in such a way that the following two conditions hold:

$$(\tilde{q}\tilde{f} - \tilde{p})(w) = \mathcal{O}(w^s) \tag{5.16a}$$

$$\tilde{p}(w)\tilde{q}(w^{-1}) + \tilde{p}(w^{-1})\tilde{q}(w) = \sum_{k=0}^{m} \alpha_k (w^k + w^{-k}), \tag{5.16b}$$

for some coefficients $\{\alpha_k\}$, where s is as large as possible. The function r is then defined as

$$r(x) := \frac{1}{2}[\tilde{r}(w) + \tilde{r}(w^{-1})], \tag{5.17}$$

$$= [\tilde{p}(w)\tilde{q}(w^{-1}) + \tilde{p}(w^{-1})\tilde{q}(w)]/2\tilde{q}(w)\tilde{q}(w^{-1}) \tag{5.18}$$

If $\tilde{r}$ doesn't have poles in the closed unit disk, it can be shown that r is a CP approximation of type $[m/n]$ for f, i.e. r is continuous, has a Chebyshev series expansion satisfying

$$f(x) - r(x) = \mathcal{O}(T_s(x)),$$

and is a rational function of type $[m/n]$. Notice that condition (5.16a) on the intermediate Laurent-Padé approximant $\tilde{r}$ is the usual Padé requirement, and condition (5.16b) enforces that the numerator of r is of degree less or equal than m (otherwise the resulting approximant would be a rational function of type $[n/n]$ in case $m < n$).

The computation of CP approximations requires first that we obtain the coefficients $\{\tilde{\alpha}_k\}$ and $\{\tilde{\beta}_k\}$ of the intermediate Laurent-Padé approximant, and then that we compute the Chebyshev coefficients of the numerator and denominator of r. For the first part we solve the usual Padé system for the Laurent series given by

$$\tilde{\alpha}_k = \sum_{i=0}^{k} \tilde{\beta}_i c_{k-i}, \quad 0 \le k \le m \tag{5.19a}$$

$$H_{m,n}\boldsymbol{\beta} = -\tilde{\beta}_0 \boldsymbol{\gamma}, \tag{5.19b}$$

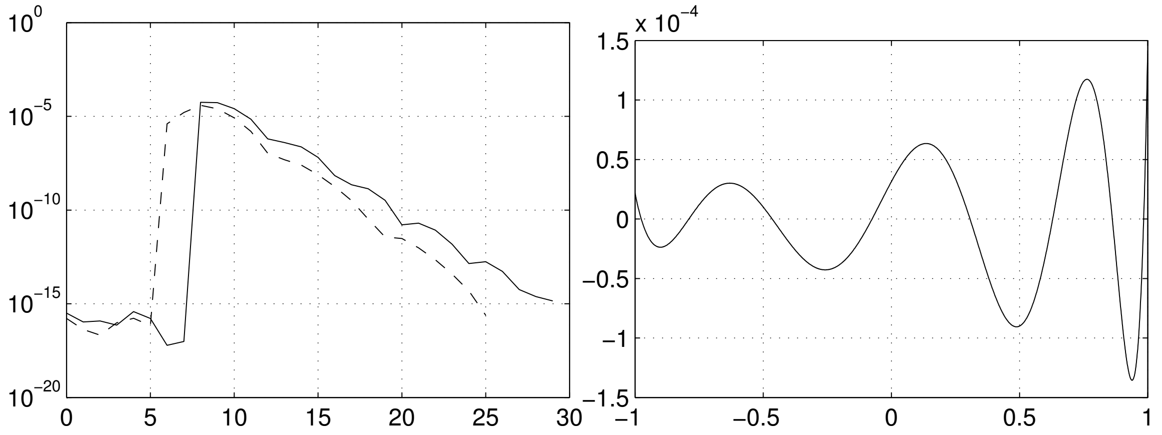

Figure 5.14: CP approximant p/q of type [5/2] to $f = \sin(\exp(x)) + \exp(\sin(x))$. Left: Chebyshev coefficients of $qf - p$ (dashed) and $f - p/q$ (solid). Right: Error $f - p/q$. The norm of the error is $1.417035506152686 \times 10^{-4}$.

where $\boldsymbol{\beta} = [\tilde{\beta}_n, \tilde{\beta}_{n-1}, \dots, \tilde{\beta}_1]^\mathsf{T}$, $\boldsymbol{\gamma} = [\gamma_{m+1}, \gamma_{m+2}, \dots, \gamma_{m+n}]^\mathsf{T}$ and $H_{m,n}$ is the order n Hankel matrix

$$\begin{pmatrix} c_{m-n+1} & c_{m-n+2} & \cdots & c_m \\ c_{m-n+2} & c_{m-n+3} & \cdots & c_{m+1} \\ \vdots & \vdots & & \vdots \\ c_m & c_{m+1} & \cdots & c_{m+n-1} \end{pmatrix},$$

where we set $c_i = c_{|i|}$ in case $i < 0$. It can be shown that the system (5.19b) always has nontrivial solutions, that each such solution defines the same rational function (5.15) satisfying conditions (5.16) and has a unique reduced representation if we set the constant coefficient $\tilde{\beta}_0 = 1$ in the denominator.

From the cross-product (5.18), the coefficients of the numerator p and denominator q of r are obtained directly from the solution of system (5.19):

$$p(x) = \sum_{k=0}^{m} \alpha_k T_k(x) := 2 \sum_{j=0}^{l} \sum_{k=0}^{n} \tilde{\alpha}_j \tilde{\beta}_k T_{|j-k|}(x) \tag{5.20a}$$

$$q(x) = \sum_{k=0}^{n} \beta_k T_k(x) := 2 {\sum_{k=0}^{n}}' T_j(x) \left\{ \sum_{k=j}^{n} \tilde{\beta}_k \tilde{\beta}_{k-j} \right\}. \tag{5.20b}$$

From condition (5.16a), it follows that the coefficients of $T_i(x)$ for $i > m$ in (5.20a) are zero.

Figure 5.14 shows the [5/2]-CP approximation of $f(x) = \sin(\exp(x)) + \exp(\sin(x))$ computed with the code `chebpade` (see Figure 5.15). Compare this figure with Figure 5.13 and note that the first 8 coefficients of the CP approximation are indeed negligible and the error in the infinity norm is smaller.

We should mention that the construction we just presented encounters a difficulty when $\tilde{r}$ has poles in the closed unit disk. Trefethen and Gutknecht [169] proposed a different

```
function [p,q] = chebpade(f,m,n)

l = max(m,n);                                 % temp degree in case m < n
c = fliplr( chebpoly(f) )'; c(1) = 2*c(1);   % Chebyshev coeffs
top  = c(abs([m-n+1:m]) + 1);                % top row of Hankel system
bot  = c([m:m+n-1]      + 1);                % bottom row of Hankel system
rhs  = c([m+1:m+n]      + 1);                % rhs of Hankel system
beta = 1;
if n > 0,
  beta = flipud([-hankel(top,bot)\rhs;1]) ;  % den coeffs of Laurent-Pade
end
c(1) = c(1)/2;
alpha = conv( c(1:l+1), beta );              % numerator of Laurent-Pade
alpha = alpha(1:l+1);
beta = beta';
D = zeros(l+1,l+1);                          % temporary matrix
D(1:l+1,1:n+1) = alpha(:,ones(n+1,1)).*...
    beta(ones(l+1,1),:);
pk(1) = sum( diag(D) );
for k = 1:m
  pk(k+1) = sum( [diag(D,k); diag(D,-k)] );  % coeffs of Cheb-Pade numerator
end
for k = 1:n+1
  u = beta(1:n+2-k); v = beta(k:end);
  qk(k) = u*v';                              % coeffs of Cheb-Pade denominator
end
pk = pk/qk(1); qk = 2*qk/qk(1); qk(1) = 1;
p = chebfun(chebpolyval(fliplr(pk)));        % chebfun of Cheb-Pade numerator
q = chebfun(chebpolyval(fliplr(qk)));        % chebfun of Cheb-Pade denominator
```

Figure 5.15: Code of the default option in the Chebyshev–Padé algorithm of Chebfun. The input arguments are a chebfun `f` and the degrees `m` and `n` of the numerator and denominator to be computed and the output arguments are chebfuns `p` and `q` of the CP approximation to `f`.

construction, based on the reduction of the problem to one of *stable Padé approximation*, which clarified this issue.

5.4 Discussion

The subject of this chapter was the study of approximation by rational functions, and we focused on two methods: interpolation and Padé-type techniques. Most of our work has been on the former, and in this chapter we considered its theoretical and computational aspects.

The direction taken along the rational interpolation table completely determines the type of analysis that can be done. In the case of the row direction, we saw that the poles of the approximant approach those of meromorphic functions, and this allows one to use the Hermite integral formula. In a sense, this is analogous to the case of rational interpolation with prescribed poles, where the Hermite integral can be modified to include that information (see [177, p. 186]). When the approximations come from the diagonal of the table, however, a whole new kit of techniques is necessary. The convergence result for rational interpolants on the diagonal to analytic functions is an extension of the Nuttall–Pommerenke theorem (see [157]). The type of convergence, however, is in *capacity*, commonly used in the analysis of diagonal rational approximants.

We have seen that for functions with singularities on the domain, rational interpolants converge at faster rates than polynomials. For example, Newman, Werner, and Brutman have established precise results for the convergence of diagonal rational interpolants in various grids to $|x|$. For interpolants to the step function, we have shown numerical evidence that row interpolants decrease the effect of the Gibbs phenomenon. It would be interesting to obtain rigorous results that explain the rate at which the overshoot and the error decreases for rational interpolation, similar in nature to those presented in [81] for the Gibbs phenomenon in Chebyshev–Padé approximation.

Rational interpolants computed in truncated arithmetic, however, differ from the exact rational interpolants. The main contribution in this chapter is to settle some of the issues that cause this discrepancy. Problems due to poles that appear in the domain of the function or the existence of unattainable points are well-known. For the later, in particular, there is a connection with the patterns formed in the rational interpolation table.

One of the motivations to use rational interpolants is the possibility of converging in grids of points for which the polynomial case diverge. For arbitrary grids, however, the evaluation with the rational barycentric formula present a new difficulty: The rational barycentric weights are obtained from polynomial barycentric weights as $u_j = w_j q(x_j)$, which in turn are sensitive to the distribution of the nodes. A method combining the approach of Berrut and Mittelman [13], which avoids such calculation, with the one presented in this thesis, perhaps could bypass this difficulty.

But even in “well-behaved” grids, like sets of Chebyshev points, for example, rational interpolants computed in IEEE arithmetic exhibit a strange behaviour. Starting with low degrees of the numerator or denominator, as one increases either or both of them, the computed rational interpolant agrees with the theory. However, when reaching a certain number of interpolation points, the interpolant suddenly deviates completely from what is expected. This phenomenon occurs even for low degrees of the numerator or denominator, or when the error is still far from machine precision.

In Section 5.2 we tracked the anomaly in the interpolant to the singular values of the matrix that describes the linear system associated to the condition (4.2). When the smaller of these singular values reach machine precision, artificial common factor appear in the complex plane, destroying the expected behaviour. The fundamental result for understanding the connection between singular values of the associated linear system and the common factors of the rational interpolant is Theorem 5.5. Figures 5.9 and 5.11 illustrate this point. Notice that the error of the computed interpolant may keep decreasing as one increases the number of points, even to machine precision. The distinction that we want to make here, however, is that the zeros of the numerator and denominator no longer agree with the theoretical ones.

A fundamental question that remains to be answered is why the singular values decrease in such a way on the first place. Future work on rational interpolation may seek to characterise the behaviour of the singular values in terms of the properties of the target function. This can be a first step in establishing methods that attempt to correct the discrepancy between computed and theoretical rational interpolants.

Padé approximation has been the focus of much research. Its construction involves the solution of a Hankel system and many of the topics we discussed for rational interpolation were in fact developed earlier for Padé approximants. The construction of Chebyshev–Padé approximants requires the previous computation of a Laurent–Padé approximant and the explicit constraining of the degree of the numerator. We have presented a Matlab code that implements these ideas, and in the next chapter we use it for the initialisation of the rational Remez algorithm.

CHAPTER 6

Algorithms for Best Polynomial and Rational Approximation[1]

In his writings, a wise Italian
says that the best is the enemy of the good

Voltaire
La Bégueule (1772)

So far in this thesis we have studied some aspects of the construction and convergence of interpolants as approximants. In particular, we saw in Section 2.3 that polynomial interpolants are "near-best" in the sense that the ratio between their error and the minimum error obtained by any polynomial of the same degree is bounded by $1 + \Lambda_{\mathbf{x}}$, where $\Lambda_{\mathbf{x}}$ is the Lebesgue constant of the interpolation grid $\mathbf{x}$ (cf. definition (2.40)). Since the Lebesgue constant for Chebyshev points grows only at a logarithmic rate (2.42), interpolants in these nodes produce approximants that are very close to optimal.

In this chapter we go a step further and explore the Remez algorithms that allow us to compute best approximations. Since these algorithms have the same starting point for both the polynomial and rational case, we state the problem in the following general form: Given a continuous function f on $[-1, 1]$, find a function $r^* \in \mathcal{R}^+(m, n)$ such that

$$\|f - r^*\| \leq \|f - r\| \quad \text{for all } r \in \mathcal{R}^+(m, n), \tag{6.1}$$

where $\|\cdot\|$ denotes, as always, the supremum norm on $[-1, 1]$ and $\mathcal{R}^+(m, n)$ is the subclass of functions in $\mathcal{R}(m, n)$ that are bounded in $[-1, 1]$. A necessary and sufficient condition for

[1]The material presented in this chapter is largely adapted from R. Pachón and L. N. Trefethen, *Barycentric-Remez algorithms for best polynomial approximation in the chebfun system*, BIT Numer. Math. (2009), 49: 721–741 [125], where the polynomial case was studied. The contribution of the author of this thesis to that paper included most of the technical developments of the algorithm, the Matlab codes, the numerical experiments, the research for some of the historical discussions, and most of the writing of the paper. Trefethen had the idea in the first place to look for an implementation of the Remez algorithm in Chebfun, and his technical contribution was fundamental to assure the robustness that allows one to work with degrees in the thousands. Moreover, he had a significant hand in the writing of the paper.

an irreducible rational function in $\mathcal{R}(m,n)$ to be in $\mathcal{R}^+(m,n)$ is that its denominator has no zeros in $[-1,1]$. For the particular case of $n=0$ we denote the best approximation by p^*. As before, we use the convention $N=m+n$. Note that in this chapter we are concerned only with approximations on a real interval. Algorithms for functions defined in the complex plane, although related to the ones presented here, require particular considerations that we do not mention (see for example [89] in which the best polynomial approximation on the unit circle is constructed from the best approximants on $[-1,1]$).

Discussions of this problem can be found in every book on approximation theory [27, 34, 93, 112, 117, 130, 138]. The existence of a best polynomial approximation follows from the existence of a best approximant in compact subsets of metric spaces (see for example [130, Thm. 1.2] or [27, Sec. 1.6]). For rational functions the proof of existence requires a different argument [27, Sec. 5.2]. The uniqueness in both cases is a consequence of the characterisation theorem that we present below (cf. Theorem 6.1 and [27, Sec. 5.4]).

Starting with Chebyshev himself, the best polynomial approximation problem was studied from the second half of the 19th century to the early 20th century, and by 1915 the main results had been established [158]. According to Cheney [27, p. 230–231], many of the results in the theory of best rational approximation were developed between 1930s and 1960s. The treatise by Walsh [177], published for the first time in 1935, is perhaps one of the first references with a comprehensive discussion on rational approximation and it includes a chapter on best rational approximation.

A new wave of interest in the computational aspects came in the 1950s and 1960s. The focus of much of this work was the algorithm introduced by Remez [135, 136, 137], and in this period a deep understanding of its theoretical properties as well as numerous variations for its practical implementation were developed. The way in which the Remez algorithm for polynomials came to the attention of researchers in the West was perhaps through a 1951 paper in Russian by Novodvorskii and Pinsker which is cited by Shenitzer in 1957 [153] and Murnaghan and Wrench in 1959 [118]. The rational case was treated as early as 1963 in the posthumous article of Maehly [98]. In the 1970s the Remez algorithm also became a fundamental tool of digital signal processing, where it was introduced by Parks and McClellan in the context of filter design [126].

Concerning software, some items of note are

- ACM Algorithm 318, by Boothroyd in 1967 [15]
- ACM Algorithm 414, by Golub and Smith in 1971 [61]
- The Remes-difcor algorithm, by Kaufman, Leeming and Taylor in 1981 [83]
- ACM Algorithm 604, by Sauer in 1983 [150]
- RATCH/DRATCH from the IMSL library [79]

- E02ACF from the NAG library [119]
- REMES and FIRPM from Matlab's Signal Processing Toolbox [109]
- MINIMAX from Maple [100]
- MiniMaxApproximation from Mathematica [108]
- REMEZ from the Sollya software package [87].

Many of these implementations go back many years and several do not solve the general best approximation problem as posed above but variants involving discrete variables or digital filtering. Overall, it is not clear that there is any widely-used program at present for computing best approximations. Thus this is a problem that is very widely known among numerical mathematicians and engineers, but without many software options to choose from for its solution.

In this chapter we present new Remez algorithms for the computation of best polynomial and rational approximations. These algorithms have two crucial features:

- Construction through polynomial and rational interpolants
- Formulation within Chebfun

In a nutshell, the algorithm consists of an iterative procedure which, at every step, modifies the grid of trial interpolants and converges quadratically to the desired approximant. As in many other optimisation routines, the proximity of the initial guess is relevant in the computation. Since we rely on polynomial and rational interpolants, the fundamental tool studied and developed in this thesis, we will use many of the ideas presented in the previous chapters to construct the algorithm and explain its performance. In particular, for best polynomials we use the barycentric Lagrange interpolation formula which we know is reliable (cf. Section 2.1.3), and for best rational functions we use the algorithm for rational interpolation introduced in Chapter 4.

Although we developed the idea of using interpolants in the Remez algorithm independently of previous work, we should mention that we found, in our research, some traces of similar ideas for this problem. Most notably, a barycentric representation was introduced by Parks and McLellan [126] for trigonometric approximations. The presentation in this chapter, however, may be new for algebraic polynomials, where the features of scaling by the logarithmic capacity of the approximation interval and the implementation of barycentric weights with formula (2.22) become crucial for robustness. For best rational approximation, Maehly [98] already acknowledges that best approximations "*can be regarded as modifications of interpolation algorithms*", however he favours a method (which he calls the Second Direct Method) that updates the locations of the zeros of the error and not their extrema.

Later, Cody and Stoer [30] and Cody, Fraser, and Hart [29] proposed to represent the rational functions as interpolating continued fractions. However, the system that we will obtain is different, being constructed from the rational interpolants of Chapter 4.

The other fundamental aspect of our algorithm is its implementation in Chebfun. The use of Chebfun is new and brings many new angles to the problem, as we shall describe. In the past, parabolic and other approximations have been used which are fast but may miss extrema. The new approach relies instead on global representations of each error curve and global zerofinding, one of the hallmarks of Chebfun.

Section 6.1 presents the classical Remez algorithm for the general case of rational approximation, which consists of two main stages in each step. In Section 6.2 we discuss the first one, the computation of a trial function through polynomial and rational interpolants. In particular we derive an analytic formula for the levelled error that might be new for the case of rational functions. The second stage consists of locating local extrema in the trial error to obtain a new trial reference. We present, in Section 6.3, an efficient approach for this computation in Chebfun. The implementation for the polynomial Remez algorithm is explained in detail in Section 6.4. The codes presented in that section make it easy to explore properties of best approximations and we also show some of these possibilities. We compare the convergence of best approximants computed with Chebfun to Jackson-type bounds for continuous and Lipschitz continuous functions. The code is then extended in Section 6.5 to the rational case, where we also discuss some of its limitations.

6.1 Overview of the Remez algorithm

Since the best approximation is unique, we can define the operator that assigns to each continuous function its best rational approximation r^* of fixed type. It is well known that this operator, although continuous, is nonlinear even for polynomial approximation (for an example see [93, p. 33]), and so we need iterative methods to compute r^*. The Remez algorithm is one such method. Other important algorithms are the differential correction algorithms, which rely on ideas of linear programming and are used to solve the discrete version of this problem [4, 83, 132, 154]. We will not discuss these methods here.

We begin our discussion of the Remez algorithm by recalling two theorems that are essential to it. The first was proved by Kirchberger in 1901 for the particular case $n = 0$ (cf. [16] [27, p. 75] [130, p. 77]) and, according to Cheney [27], by Achieser in 1930 for general rational functions.

Theorem 6.1 (Equioscillation Property)**.** *A rational function $r \in \mathcal{R}^+(m,n)$ is the best approximation to f (that is, $r = r^*$) if and only if there exists a set of $N + 2 - \delta$ distinct points $\mathbf{a}^* = [a_0^*, \ldots, a_{N+1-\delta}^*]^\mathsf{T}$ in $[-1, 1]$ such that*

$$f(a_i^*) - r(a_i^*) = \lambda\sigma_i\|f - r^*\|, \quad i = 0, \ldots, N + 1 - \delta, \tag{6.2}$$

where $\sigma_i := (-1)^i$ and $\lambda = 1$ or $\lambda = -1$ is fixed. The nonnegative integer δ is the defect of the rational function r as defined in Theorem 5.5.

A set of points $\mathbf{a}^*$ that satisfies (6.2) is called a *reference.* Figure 6.1 shows p^* for $f(x) = \sin(3\pi x)\exp(x)$, $[-1, 1]$ and $N = m = 2, 5, 7$, and 10, and the references of 4, 7, 9 and 12 points respectively where $f - p^*$ equioscillates.

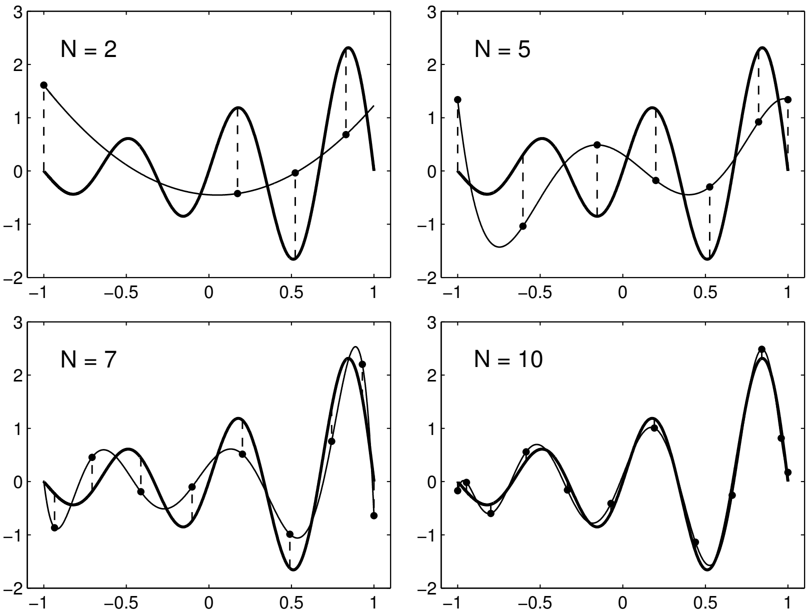

Figure 6.1: Best polynomial approximations (thin lines) of degrees 2, 5, 7 and 10 to $f(x) = \sin(3\pi x)\exp(x)$ (bold lines) on $[-1, 1]$. The dots show the polynomial at the reference and the dashed vertical bars the corresponding errors, of equal lengths and alternating in orientation.

For the function $f = \tanh(50x)$ in $[-1, 1]$, Figure 6.2 shows the best rational approximants of type [1/1], [2/2], [3/3], and [4/4] and the references of 4, 6, 8, and 10 points where the error equioscillates. For this last function Figure 6.3 shows error curves with 10 equioscillations with best approximants of type [8/0], [6/2], [4/4], and [2/6].

Theorem 6.1 can be generalised to rational approximations that satisfy the "Haar condition" [27, p. 159]. This allows us to look for best approximations in other sets of functions, for example quotients of trigonometric polynomials. Analogous equioscillation properties hold for best CF and Padé approximations [165].

The second theorem, proved by de la Vallée Poussin in 1910 [36], establishes an inequality between the alternating error of a trial rational function and the error of the best approximation [27, p. 77][112, Thm. 64, 98][130, Thm. 7.7].

Theorem 6.2 (de la Vallée Poussin)**.** *Let $r \in \mathcal{R}^+(m, n)$ and $\mathbf{a} = [a_0, \ldots, a_{N+1}]^\mathsf{T}$ be a set of $N + 2 - \delta$ distinct points in $[-1, 1]$ such that* $\operatorname{sign}\big(f(a_i) - r(a_i)\big) = \lambda\sigma_i$, $i = 0, \ldots, N + 1 - \delta$,

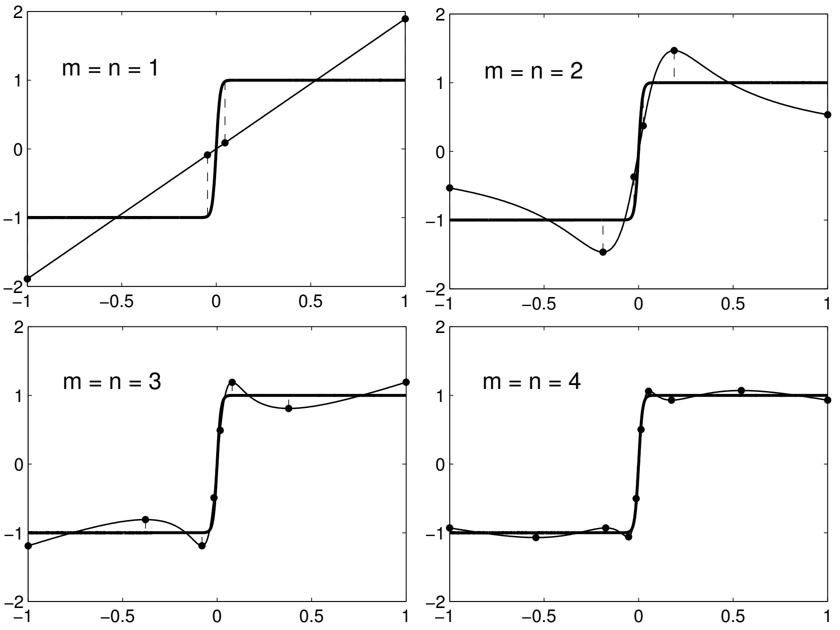

Figure 6.2: Best rational approximations (thin lines) of type [1/1], [2/2], [3/3] and [4/4] to $f(x) = \tanh(50x)$ (bold lines) on $[-1, 1]$ (conventions analogous to Figure 6.1).

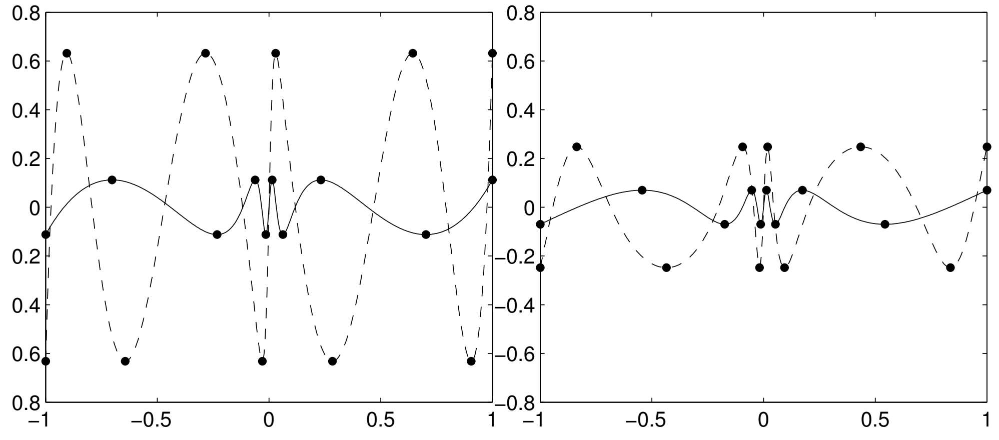

Figure 6.3: Errors of best rational approximants of types [8/0] (left-dashed), [6/2] (left-solid), [4/4] (right-solid), and [2/6] (right-dashed) to $f(x) = \tanh(50x)$ on $[-1, 1]$. In each case the error equioscillates 10 times in the locations marked by the dots. The maximum errors are approximately 0.632147, 0.112227, 0.069968, and 0.247887 respectively.

with σ_i and λ defined as in Theorem 6.1. Then, for every $s \in \mathcal{R}^+(m, n)$,

$$\min_i |f(a_i) - r(a_i)| \leq \max_i |f(a_i) - s(a_i)|, \tag{6.3}$$

and in particular

$$\min_i |f(a_i) - r(a_i)| \leq \|f - r^*\| \leq \|f - r\|. \tag{6.4}$$

Theorem 6.2 asserts that a rational function $r \in \mathcal{R}^+(m,n)$ whose error oscillates $N+2-\delta$ times is "near-best" in the sense that

$$\|f-r\| \leq C\|f-r^*\|, \qquad C = \frac{\|f-r\|}{\min_i |f(a_i)-r(a_i)|} \geq 1. \tag{6.5}$$

The Remez algorithm constructs a sequence of trial references $\{\mathbf{a}^k\}$ and trial rational functions $\{r_k\}$, $k = 0,1,\dots, r_k \in \mathcal{R}^+(m,n)$, that satisfy this alternation condition assuming that $\delta = 0$ and in such a way that $C \to 1$ as $k \to \infty$ (if $\delta \neq 0$ special considerations have to be made, see the discussion in p. 113). At the kth step the algorithm starts with a trial reference $\mathbf{a}^k$ and then computes a rational function r_k such that

$$f(a_i) - r_k(a_i) = \sigma_i h_k, \quad i = 0,\dots,N+1-\delta, \tag{6.6}$$

where h_k is the *levelled error* (positive or negative). Then, a new trial reference $\mathbf{a}^{k+1}$ is computed from the extrema of $f - r_k$ in such a way that $|h_{k+1}| \geq |h_k|$ is guaranteed. This monotonic increase of the levelled error is the key observation in showing that the algorithm converges to r^* (see [130, Thm. 9.3]). See Figure 6.4 for the first iterations of the Remez algorithm in the computation of the best approximant of type [5/5] to the gamma function $\Gamma(x)$, $x = [0.01, 6]$.

In Section 6.2 we explain how to compute a trial rational function and levelled error from a given trial reference, and in Section 6.3 we show how to adjust the trial reference from the error of the trial rational function.

6.2 Interpolation and best approximation: From a reference to a trial function

We discuss the first crucial feature of the proposed algorithm: the role of polynomial and rational interpolants in the construction of best approximants. We drop the subscripts and superscripts that indicate the iteration of the Remez algorithm, since we only refer to one reference throughout this section.

First we consider the polynomial case $N = m$, $n = 0$. Let $\{\phi_j\}$, $j = 0,\dots,N$ be a basis of $\mathcal{P}(N)$ and express the elements of the latter in the form

$$p(x) = \sum_{j=0}^{N} c_j \phi_j(x).$$

A continuous function f and a set $\mathbf{a} = [a_0,\dots,a_{N+1}]^{\mathsf{T}}$ of $N+2$ points uniquely determine a polynomial p and a levelled error h such that (6.6) is satisfied. The conditions (6.6) amount to a linear system of $N+2$ equations in $N+2$ unknowns: $N+1$ parameters to describe

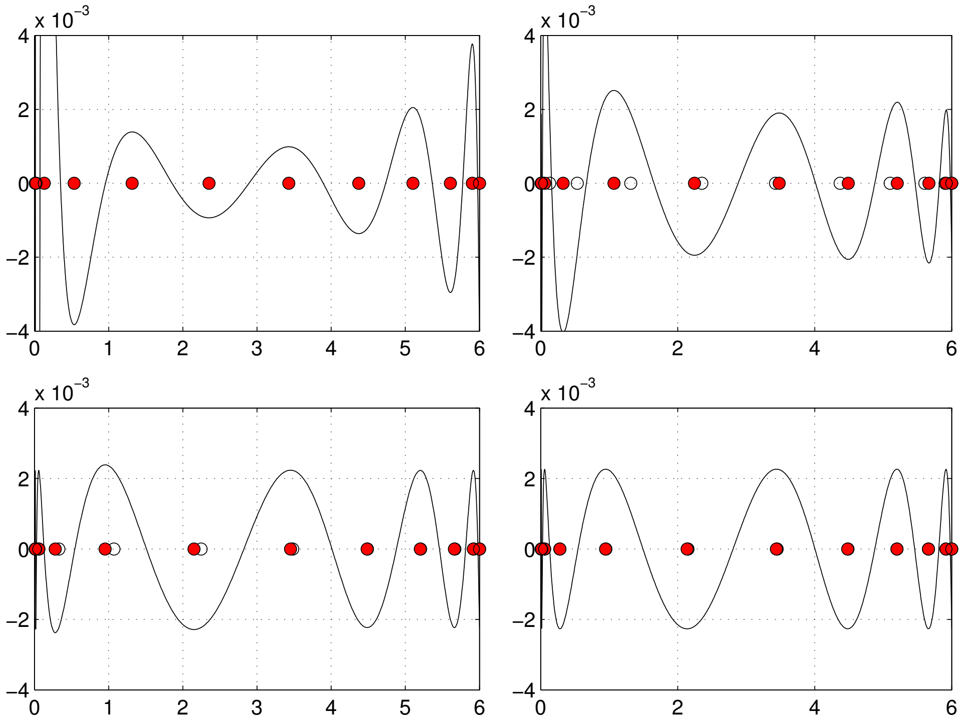

Figure 6.4: First four steps of the Remez algorithm for the best approximation of type [5/5] to $\Gamma(x)$, $x = [0.01, 6]$. The top-left panel shows the error of the initial guess, obtained by the Chebyshev–Padé algorithm of Chapter 5. The other panels show the error in the first (top-right), second (bottom-left), and third (bottom-right) iterations. The red dots show the alternating reference used to construct the trial function in the next step (the white circles show the previous reference). In all the figures there are 12 points in the reference, and some of them are clustered to the left.

the polynomial, plus the unknown h:

$$\begin{pmatrix} \phi_0(a_0) & \phi_1(a_0) & \cdots & \phi_N(a_0) \\ \phi_0(a_1) & \phi_1(a_1) & \cdots & \phi_N(a_1) \\ \vdots & \vdots & & \vdots \\ \phi_0(a_n) & \phi_1(a_n) & \cdots & \phi_N(a_N) \\ \phi_0(a_{N+1}) & \phi_1(a_{N+1}) & \cdots & \phi_N(a_{N+1}) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_N \end{pmatrix} = \begin{pmatrix} f(a_0) + \sigma_0 h \\ f(a_1) + \sigma_1 h \\ \vdots \\ f(a_N) + \sigma_N h \\ f(a_{N+1}) + \sigma_{N+1} h \end{pmatrix}. \qquad (6.7)$$

As we already saw in Section 2.1, the basis used to represent $\mathcal{P}(N)$ determines the numerical solution of a Vandermonde-type system like (6.7). For the Remez algorithm, the monomials and the Chebyshev basis seem to be the two most frequent choices for representing polynomials, an observation also made by Dunham [38]. In [125] we proposed instead to use the Lagrange basis and evaluate the trial polynomials with the barycentric formula (2.21). Although it is a simple alternative, we did not find it in the numerical analysis literature related to the Remez algorithm, and its explicit use is only mentioned in digital filter design [126].

From a reference $\mathbf{a}$ consisting of $N + 2$ points, we choose a subset $\hat{\mathbf{a}}$ of $N + 1$ points and form with them the basis $\{\phi_j\}$ of $N+1$ Lagrange polynomials defined by (2.16). It follows that the matrix of (6.7) is the $(N+1)\times(N+1)$ identity with the exception of an additional jth row whose entries are the values of the various Lagrange functions at the point a_j, which is in $\mathbf{a}$ but not in $\hat{\mathbf{a}}$. Discarding this row, we end up with the system

$$p(a_i) = f(a_i) + \sigma_i h, \quad i = 0,\ldots,n+1,\ i \neq j. \tag{6.8}$$

We will show in a moment an analytic expression for the levelled error h independent of the values $p(a_i)$, $i = 0,\ldots,N+1$ (cf. 6.13). The values on the right of (6.8) can be computed a priori at each step, defining a standard interpolation problem which determines the trial polynomial p. Since we do not have to solve the system (6.7) but only to compute the values $p(a_i)$ from (6.8), ill-conditioning is not a problem, which may be the case when working with other bases for $\mathcal{P}(N)$, even with Chebyshev polynomials (cf. Section 2.1.3).

Having discussed in detail the construction of a trial function from a reference for polynomials, we present now the derivation for rational functions. System (6.6) can be interpreted as the problem of obtaining a rational interpolant of the function g defined on a given reference $\mathbf{a}$ of $N+2$ points by $g(a_i) = f(a_i) - \sigma_i h$ (this is what Maehly calls "*interpolation with weighted deviations*" [98]):

$$r(a_i) = g(a_i) = f(a_i) - \sigma_i h, \quad i = 0,\ldots,N+1, \tag{6.9}$$

where $r = p/q \in \mathcal{R}^{+}(m,n)$ and $p \in \mathcal{P}(m)$ and $q \in \mathcal{P}(n)$. We construct such a rational interpolant with the method described in Chapter 4. Suppose that the polynomials $\{\phi_j\}$, $j = 0,\ldots,N$, are orthogonal with respect to the inner product $\langle\cdot,\cdot\rangle_{\mathbf{a}}$ defined by (4.11), and we construct the Vandermonde-type matrix $C = [\phi_0(\mathbf{a})|\cdots|\phi_{N+1}(\mathbf{a})]$ of size $(N+2)\times(N+2)$. Then, following the same argument as in Theorem 4.1, we have that the coefficients $\boldsymbol{\beta} = [\beta_0,\ldots,\beta_n]^{\mathsf{T}}$ of the denominator q of r are such that

$$\boldsymbol{\beta} \in \ker(C^{*}_{m+1:N+1}\Psi C_{0:n}), \tag{6.10}$$

where Ψ is a $(N+2)\times(N+2)$ diagonal matrix with (j,j) entry $\Psi_j = g(a_i) = f(a_i) - \sigma_i h$. The difference between this condition and the one in Theorem 4.1 is that the left matrix $C^{*}_{m+1:N+1}$ in (6.10) has $(n+1)$ rows instead of just n. Since we have a grid of $N+2$ points, we need an additional condition to cancel out the $(N+1)$-st coefficient of the numerator of r. Thus, we can formulate the condition on $\boldsymbol{\beta}$ in (6.10) as a solution of the following generalised eigenvalue problem of size $n+1$:

$$\left[C^{*}_{m+1:N+1}\Phi C_{0:n}\right]\boldsymbol{\beta} = h\left[C^{*}_{m+1:N+1}\Delta C_{0:n}\right]\boldsymbol{\beta}, \tag{6.11}$$

where Φ is the usual diagonal matrix with entries $\Phi_j = f(a_j)$ and Δ is a diagonal matrix with (j,j) entry $\Delta_j = \sigma_j$, i.e. $\Psi = \Phi - h\Delta$.

Each eigenpair of system (6.11) defines a levelled error and denominator of r, however only one determines a bounded rational function in $[-1,1]$ (see [130, p. 115]). The coefficients $\boldsymbol{\beta}$ of this one can then be used to obtain the values $q(\mathbf{a})$ required to evaluate the rational function r in the rational barycentric form (4.9) (cf. Section 4.1.2).

Although the levelled error can be expressed independently of the values of the trial function in the polynomial case, the same does not hold for general rational functions. From (6.9) and since $r \in \mathcal{R}^+(m,n)$, we can pick an arbitrary point a_j from $\mathbf{a}$ and interpolate the value $r(a_j)$ exactly with an interpolant $\hat{r} \in \mathcal{R}^+(m,n)$ on a subset grid $\hat{\mathbf{a}}$ of $N+1$ points of $\mathbf{a}$ that omits the point a_j. The rational barycentric form of $\hat{r}$ is

$$\hat{r}(x) = \sum_{\substack{i=0\\ i\neq j}}^{N+1} \frac{\hat{w}_i q(a_i) g(a_i)}{x-a_i} \Bigg/ \sum_{\substack{i=0\\ i\neq j}}^{N+1} \frac{\hat{w}_i q(a_i)}{x-a_i} \quad \text{with} \quad \hat{w}_i = \frac{1}{\prod\limits_{k\neq i,j}(a_j - a_k)}.$$

Evaluating $\hat{r}$ at a_j we obtain

$$\hat{r}(a_j) = \sum_{\substack{i=0\\ i\neq j}}^{N+1} \frac{\hat{w}_i q(a_i) g(a_i)}{a_j-a_i} \Bigg/ \sum_{\substack{i=0\\ i\neq j}}^{N+1} \frac{\hat{w}_i q(a_i)}{a_j-a_i} = \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i) g(a_i) \Bigg/ \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i),$$

where $\{w_i\}$ are the barycentric weights we would obtain if we considered all the $N+2$ points of $\mathbf{a}$ as the Lagrange nodes. Since $\hat{r}(a_j) = r(a_j) = g(a_j) = f(a_j) - \sigma_j h$, it follows that

$$\sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i) g(a_i) = f(a_j) \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i) - \sigma_j h \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i).$$

Summing over j and noting that $q(a_i)g(a_i) = p(a_i)$, we obtain

$$\sum_{j=0}^{N+1} \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i p(a_i) = \sum_{j=0}^{N+1} f(a_j) \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i) - h \sum_{j=0}^{N+1} \sigma_j \sum_{\substack{i=0\\ i\neq j}}^{N+1} w_i q(a_i)$$

which by rearranging terms becomes

$$(N+1) \sum_{i=0}^{N+1} w_i p(a_i) = \Big(\sum_{i=0}^{N+1} f(a_i) + h\sigma_i\Big)\Big(\sum_{i=0}^{N+1} w_i q(a_i)\Big) - \sum_{i=0}^{N+1} f(a_i) w_i q(a_i) + h \sum_{i=0}^{N+1} \sigma_i w_i q(a_i).$$

We now use the identity (2.24) derived in Section 2.1.3. Since p and q are polynomials of degrees strictly less than $N+1$, the sum on the left and the first term on the right of the previous equation are zero. We therefore obtain the following formula for the levelled error:

$$h = \frac{\sum\limits_{j=0}^{N+1} f(a_j) w_j q(a_j)}{\sum\limits_{j=0}^{N+1} \sigma_j w_j q(a_j)}. \tag{6.12}$$

If $n = 0$ then $q(a_i) = 1$, $i = 0, \dots, N+1$, i.e. the denominator is constant and the levelled error is

$$h = \frac{\sum_{j=0}^{N+1} f(a_j) w_j}{\sum_{j=0}^{N+1} \sigma_j w_j}, \tag{6.13}$$

which shows that h is independent of the trial function. On the other hand, if $n > 1$, the levelled error is intrinsically defined by the values of the denominator of r, and we cannot hope to decouple h from the system (6.11).

6.3 Location of extrema: From a trial function to a reference

Having computed a trial function from a reference, the second stage at each Remez step consists of computing a new reference from the trial function. Suppose that for a trial reference $\mathbf{a}^k$ there is a rational function r_k such that $f(a_i^k) - r_k(a_i^k) = \sigma_i h_k$ but $|h_k| < \|f - r^*\|$. The goal is to obtain a new reference $\mathbf{a}^{k+1} = [a_0^{k+1}, \dots, a_{N+1}^{k+1}]^\mathsf{T}$ where the error of $r_{k+1} \in \mathcal{R}^+(m,n)$ equioscillates with the levelled error $|h_{k+1}| > |h_k|$.

The crucial condition to replace $\mathbf{a}^k$ by $\mathbf{a}^{k+1}$ is that r_k oscillates on $\mathbf{a}^{k+1}$ (but not necessarily equioscillates) with amplitude greater than or equal to than $|h_k|$, i.e.

$$|h_k| = |f(a_i^k) - r_k(a_i^k)| \le \min_i |f(a_i^{k+1}) - r_k(a_i^{k+1})|, \quad i = 0, \dots, N. \tag{6.14}$$

Since, as shown in the previous section, the construction of r_{k+1} is such that the error equioscillates on $\mathbf{a}^{k+1}$ with amplitude $|h_{k+1}|$, it follows from Theorem 6.2 that this condition implies the increase of the levelled error at the next iteration (r_k and r_{k+1} being the functions r and s respectively in Theorem 6.2).

Remez [136] proposed two strategies to achieve this. One is to move one of the points of $\mathbf{a}^k$ to the abscissa of the global extremum while keeping the sign alternation; the other is to replace all the points of $\mathbf{a}^k$ by $N+2$ oscillating local extrema satisfying (6.14) and to include in $\mathbf{a}^{k+1}$ the abscissa of the global extremum. These strategies are known as the first and second Remez algorithms, respectively.

More specifically, the first Remez algorithm constructs $\mathbf{a}^{k+1}$ by exchanging a point a_old in $\mathbf{a}^k$ with the global extremum a_new of $f - r_k$ in such a way that the alternation of signs of the error is maintained. If $a_0 < a_\text{new} < a_{N+1}$, then a_old is the closest point in $\mathbf{a}^k$ for which the error has the same sign as at a_new. If $a_\text{new} < a_0$ and the signs of a_new and a_0 coincide then a_old is a_0; if $a_\text{new} < a_0$ but the signs of a_new and a_0 are different, then a_old is a_{n+1}. Similar rules apply if $a_\text{new} > a_{N+1}$.

The second Remez algorithm constructs the set $\tilde{\mathbf{a}}^{k+1}$ of points in $\mathbf{a}^k$ and local extrema $\{b_i\}$ of $f - r_k$ such that $|(f - r_k)(b_i)| > |h_k|$. Then, for each subset of $\tilde{\mathbf{a}}^{k+1}$ of consecutive points with the same sign it keeps only one for which $|f - r_k|$ attains the largest value. From the resulting set, $\mathbf{a}^{k+1}$ is obtained by choosing $N+2$ consecutive points that include the global extremum of $f - r_k$.

Assuming f is twice differentiable, the error of the Remez algorithm decays at a quadratic rate if we the measure error at every $N+2$ steps in the case of the first Remez algorithm [130, Sec. 9.4] and at every step in the case of the second [176]. The convergence properties for the algorithm in the rational case are similar to those for polynomials [130, Sec. 10.3].

This problem of updating the trial reference is an optimisation problem: for the error function $f(x) - r_k(x)$, find the global extremum or the alternating local extrema. In the context of Remez algorithms, a standard technique for computing these extrema [35] consists of constructing a parabola for each node a_k in the reference that interpolates the error at a_k and two other consecutive points. The extremum of the parabola is then exchanged with the closest of the three values and the process starts again. Golub and Smith [61], for example, use this strategy in their Remez algorithm. Though they claim that the old nodes in the reference provide a good initial guess for the extrema, they acknowledge the possibility that the parabola method may not yield acceptable values, in which case they switch to a crude search method. They write *"most of the programming effort is involved in locating the extrema of the error function"* [61].

This is where the Remez algorithm implementation in Chebfun brings a novel alternative. From the chebfun `f` of the target function f we can construct the chebfun of the error. In the polynomial case, the trial function can be represented by a chebfun `p`, and to compute the local extrema of `e = f - p` we use the efficient rootfinder `roots` of Chebfun, mentioned in Section 3.4. In the rational case we have to be careful because the trial function may not be efficiently represented by a chebfun—indeed this will almost by definition be difficult in any case where rational approximations are much better than polynomial. Instead, we obtain the local extrema of the error from the zeros of `(q.^2.*diff(f)-q.*diff(p)+p.*diff(q))`, where `p` and `q` are the chebfuns of the numerator and denominator of the trial rational function. The subfunction `exchange` in Figure 6.5 is a compact implementation of the exchange procedure just described, for the polynomial and rational case, and for the first and second algorithms.

In Chebfun, locating the global extremum of the error function requires the same computational effort as locating all the local extrema, both of them using the `roots` command. Hence, our implementation of the second Remez algorithm is usually much faster than that of the first algorithm, which usually needs many more iterations.

6.4 Implementation for best polynomial approximation

In this section we present and discuss the command `remez` in Figure 6.6, which has been included in Chebfun since version 2.0501. It produces the chebfun `p` of the best polynomial

```
function [xk,norme] = exchange(xk, h, method, f, p, varargin)
if nargin == 5                              % polynomial case
    e = f - p;                              %    error
    [~,rr] = maxandmin(e,'local');          %    extrema of the error
else
    q = varargin{1};                        % rational case
    q2de =  (q.^2).*diff(f) - ...           % q^2 * diff(e)
                q.*diff(p) + p.*diff(q);
    [a,b] = domain(f);
    rr = [a; roots(q2de); b];               %    extrema of the error
    e = @(x) f(x) - p(x)./q(x);             %    fnc handle of error
end
if method == 1                              % one-point exchange
    [~,pos] = max(abs(feval(e,rr))); pos = pos(1);
else                                        % full exchange
    pos = find(abs(feval(e,rr))>=abs(h)); % vals above levelled error
end
[r,m] = sort([rr(pos); xk]);
if isempty(xk)
    Npts = varargin{2};
else
    Npts = length(xk);
end
er = [feval(e,rr(pos));(-1).^(0:Npts-1)'*h];
er = er(m);
repeated = diff(r)==0;
r(repeated) = []; er(repeated) = [];      % delete repeated pts
s = r(1); es = er(1);                       % pts and vals to be kept
for i = 2:length(r)
  if sign(er(i)) == sign(es(end)) &&...     % from adjacent pts w/ same sign
          abs(er(i))>abs(es(end))           % keep the one w/ largest val
      s(end) = r(i); es(end) = er(i);
  elseif sign(er(i)) ~= sign(es(end))       % if sign changes, concatenate
      s = [s; r(i)]; es = [es; er(i)];      % pts and vals
  end
end
[norme,idx] = max(abs(es));                 % choose n+2 consecutive pts
d = max(idx-Npts+1,1);                      % that include max of error
xk = s(d:d+Npts-1);
```

Figure 6.5: Code of the exchange algorithm in Chebfun. The input arguments are a column vector `xk` with the trial reference $\mathbf{a}^k$, the levelled error `h` and a number `method`, with values 1 or 2 prescribing the use of the first or the second Remez algorithm respectively, the chebfun `f` of the target function f, and chebfuns `p` and `q` of the numerator and denominator of the trial function (only the former in the polynomial case). The output arguments are the modified vector `xk` with the new trial reference $\mathbf{a}^{k+1}$ and the norm `norme` of the new associated error. Notice how in the rational case we do not form a chebfun of the error but instead construct the chebfun of $(q^2f' - p'q + pq')$.

approximation of degree `N` of a chebfun `f`, which in this framework is a polynomial or a piecewise polynomial representing a target function.

The numbers [1]–[10] in the code refer to the following aspects that we consider funda-

```
function [p,err] = remez(f,N);              % compute deg N BA to chebfun f

[a,b] = domain(f);                          % endpoints of f
sigma = (-1).^[0:N+1]';                     % alternating signs
normf = norm(f); delta = normf;             % initialise variables
deltamin = delta; iter = 1;                 % initialise variables
xk = chebpts(N+2,[a,b]); xo = xk;           % initial reference           [1]
while (delta/normf > 1e-14) & iter <= 15    % main iteration              [2]
  fk = feval(f,xk);                         % function values
  w = bary_weights(xk);                     % compute barycentric weights  [3]
  h = (w'*fk)/(w'*sigma);                   % levelled reference error     [4]
  if h==0, h = 1e-15; end                   % perturb error if necessary   [5]
  pk = fk - h*sigma;                        % poly vals in the reference   [6]
  p=chebfun(@(x)bary(x,pk,xk,w),[a,b],N+1); % chebfun of trial poly        [7]
  [xk,err] = exchange(xk,h,2,f,p);          % replace reference
  if err/normf > 1e5                        % if overshoot, recompute      [8]
      [xk,err] = exchange(xo,h,1,f,p);      %     with one-point exchange
  end
  xo = xk;                                  % copy of xk if overshoot later
  delta = err - abs(h);                     % stopping value               [9]
  if delta < deltamin,                      % store poly with minimal norm [10]
    deltamin = delta; pmin = p; errmin = err;
  end
  iter = iter + 1;                          % increase counter
end
p = pmin; err = errmin;                     % return minimum error and poly
```

Figure 6.6: Code of the Remez algorithm for best polynomial approximation in Chebfun (slightly but only slightly simplified). The input arguments are a chebfun `f` and the degree `N` of the polynomial to be computed and the output arguments are a chebfun `p` of the best polynomial approximation to `f` and the error `err`.

mental for a robust implementation of the Remez algorithm.

1. *Initial reference*: Since the Remez algorithm is nonlinear, the question arises as to a suitable initial guess of the reference, i.e. $\mathbf{a}^0$. The `remez` code relies on the initial guess that has been the standard choice throughout the history of the Remez algorithm: the Chebyshev points (2.9) [118].

2. *Stopping criteria*: The main iteration of `remez` has two stopping criteria: The difference `delta` between the levelled error and the maximum error of the trial polynomial (cf. item 9 below) and the number of iterations. For most of the functions we have tried, and with moderately high `N` (say a couple of hundred), the parameter `delta` may decrease to the order of 10^{-13} in no more than 10 iterations. Nevertheless, for specific experiments involving high precision or a large `N`, it should be adjusted.

3. *Barycentric weights*: An important feature of our algorithm is that it scales well even when working in arbitrary intervals. This is accomplished by accurately computing the

barycentric weights on the trial reference which are required to obtain the levelled error and the chebfun of the trial polynomial. As we saw in Section 2.1.3, the barycentric weights must be scaled by the capacity of the interval and we must use formula (2.22) to compute the product. Both ideas are implemented in the command `bary_weights` and are crucial for the high degree computations that we show later.

4. *Levelled error*: The levelled error is computed explicitly by the formula (6.13). Note that it can be positive or negative.

5. *Perturbation of levelled error*: It may happen that the computed levelled error is below machine precision in the first iteration. In this case the error will not equioscillate on the initial reference and all the values $p(a_i)$ will be close to zero. In this case we set the levelled error to a small value and the next reference will be computed by the exchange rules of the second Remez algorithm that were explained above.

6. *Values of the trial polynomial*: With the levelled error already computed, the values of the trial polynomial are obtained as $p(a_i) = f(a_i) - \sigma_i h$, $i = 0, \ldots, N+1$. This is fundamentally different from other implementations of the Remez algorithm, where an explicit construction of the system (6.7) is required.

7. *Chebfun of trial polynomial*: The trial polynomial is computed as the interpolant in `a` using the barycentric interpolation formula through the command `bary` (cf. Section 2.1.3). Alternatively we could have used the Chebfun command `interp1`, but it would compute again the barycentric weights that we have already obtained.

   Notice that we are computing a polynomial interpolant in an arbitrary grid, and from the discussion in Chapter 2 it follows that this might be an ill-conditioned computation. In that case, the error function $f - p_k$ might not be computed accurately, which in turn would prevent the correct location of the oscillating extrema. The result would then be that the trial references cannot be computed with the required precision to increase the levelled error above a certain value and therefore `delta` stagnates above machine precision.

8. *Overshoot of the error*: We have seen that for certain functions and values of `N`, one of the steps in the second Remez algorithm constructs a trial polynomial with a very large extremum, usually located near the endpoints. The large norm of the polynomial introduces a very large error in the levelled error, breaking the computation. A simple way to solve this is to reverse the last step and recompute the trial reference, yet using the one-point exchange of the first Remez algorithm. We have seen in our experiments that this strategy usually solves the problem.

9. *Computation of tolerance*: At each step of the Remez algorithm, the levelled error

increases, converging to $\|f - p^*\|$ from below. The tolerance `delta`, computed as$\|f - p_k\| - |h|$ and used as stopping criteria (cf. item 1 above), decreases towards zero. From Theorem 6.1 it follows that this difference is zero if and only if the reference corresponds to the optimal one.

10. *Storage of trial polynomial*: If the levelled error does not increase above a certain value (cf. item 7 above), the Remez algorithm will continue until reaching the maximum number of iterations without improving the accuracy of the approximant. In that case we have found it useful to keep the trial polynomial with minimum error obtained throughout the whole computation, although it may not achieve the expected tolerance.

The generality of the function representation in Chebfun makes it possible to compute best approximations to various functions in a fraction of a second. To illustrate this point we compute $p^* \in \mathcal{P}_{10}$ for the functions in Table 6.1. The approximations are plotted in Figure 6.7.

i	f_i	$\|f_i - p_0\|$	$\|f_i - p^*\|$
1	$\tanh(x+0.5) - \tanh(x-0.5)$	0.00000058780531	0.00000030009195
2	$\sin(\exp(x))$	0.00000386118470	0.00000178623400
3	$\sqrt{x+1}$	0.04212512276261	0.01978007008380
4	$\sqrt{\lvert x-0.1\rvert}$	0.30512512446096	0.11467954016268
5	$1 - \sin(5\lvert x-0.5\rvert)$	0.40947166876230	0.14320591977421
6	$\min\{\mathrm{sech}(3\sin(10x)), \sin(9x)\}$	0.71216404197963	0.33561414233366
7	$\max\{\sin(20x), \exp(x-1)\}$	0.77453305461326	0.38723296760148
8	$\mathrm{sech}\big(10(0.5x+0.3)\big)^2 +$ $\mathrm{sech}\big(100(0.5x+0.1)\big)^4 +$ $\mathrm{sech}\big(1000(0.5x-0.1)\big)^6$	1.08706818322313	0.49987078860783
9	$\log(1.0001+x)$	2.98370118052234	1.40439492981387

Table 6.1: Errors of the initial guess p_0 and the best polynomial approximation for nine functions with $N = 10$.

Table 6.1 also includes the error of the polynomial interpolant p_0 through 11 Chebyshev points, which served as the initial trial polynomial for the Remez algorithm, and the error of the best approximation p^*. Although the former is obviously always larger than the latter, we can see the "near-best" property of Chebyshev interpolants. From (2.42), it follows that the improvement due to the Remez algorithm is less than a factor of three, and in general, for any continuous function and $N = 10$, it will always be less than a factor of 3.47.

In the Introduction we posed the question: How large can we make N when computing $p^* \in \mathcal{P}(N)$ for a given continuous function? The answer has varied with the years. In one

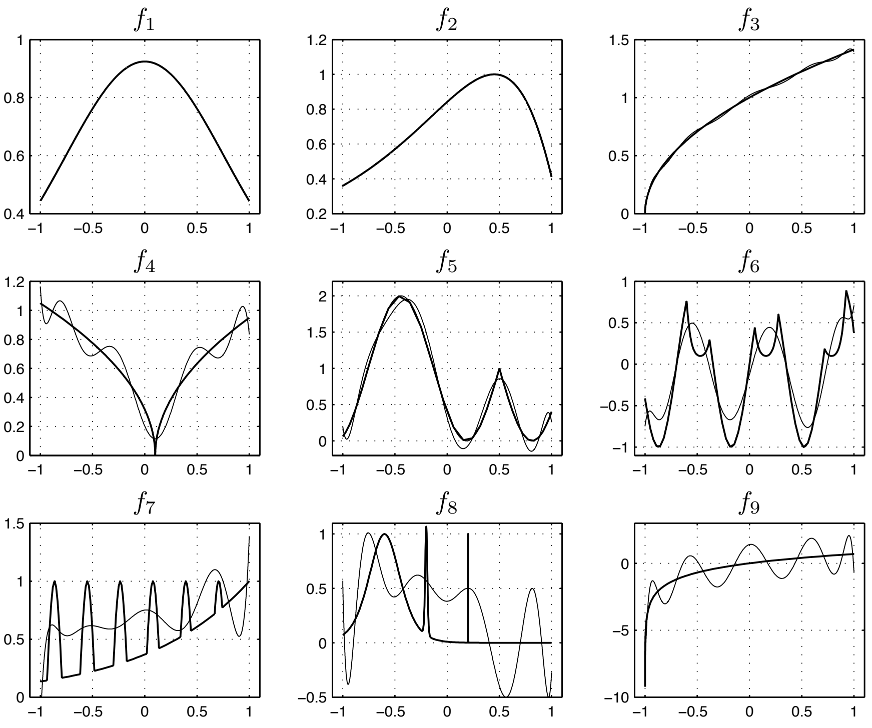

Figure 6.7: Best polynomial approximations of degree 10 (thin lines) to the functions in Table 6.1 (bold lines).

of the papers that introduced his algorithm in 1934, Remez gave the polynomial coefficients of p^* for $f(x) = |x|$ for $N = 5, 7, 9, 11$ [136]. Twenty-five years later, Stiefel [159], Curtis and Frank [32] and Murnaghan and Wrench [118] applied different techniques to compute best approximations of $\sin^{-1} x$, $\tan^{-1} x$, $\log x$, 2^x and $|x^5|$ by polynomials of degrees varying between 2 and 18.

The first computer programs for uniform approximation appeared in the 1960s, and included an Algol code by Golub and Smith [61] and Fortran codes by Barrodale and Phillips [4], and by Simpson [154] based on Schmitt's algorithm [151]. They have been used, for example, for Chebyshev curve fitting with $N > 20$. More recently, Le Bailly and Thiran [89] reported the computation of best approximants of degrees up to 64 as a step to obtaining best approximants on the unit circle in the complex plane. And higher degree approximations have been computed for particular kinds of functions. Most remarkably for its era, McClellan and Parks [110] comment on experiments which they did thirty years ago in their work with Rabiner [111] involving polynomials of degree about 500 in the context of filter design:

"From our perspective at Rice, it seemed that Larry wanted to set records for the longest optimal filter ever designed. One day we received a printout of the

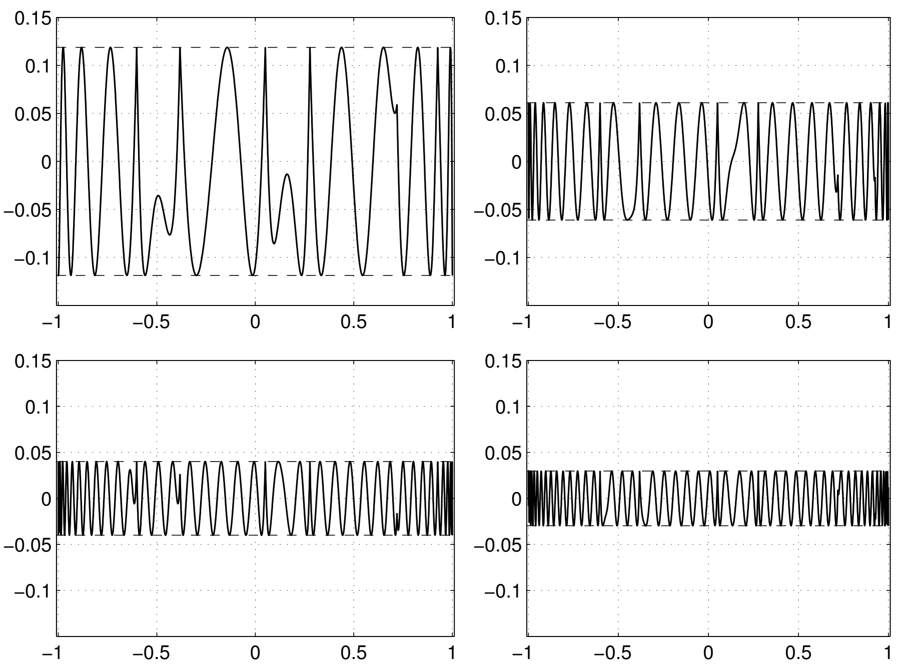

Figure 6.8: Errors of the best polynomial approximants of degrees 25, 50, 75, and 100 to $f_6 = \min\{\operatorname{sech}(3\sin(10x)), \sin(9x)\}$ in $[-1,1]$ (cf. Figure 6.7). Each computation took less than a second.

*coefficients of a length-*1401 *filter; this probably would have consumed several days of CPU time on our batch machine at Rice."*

With the Remez algorithm presented in this section we have computed best polynomial approximations with N in the hundreds and thousands in seconds or at most minutes. See Figures 6.8 and 6.9 for approximations of the function f_6 with polynomials of higher degrees.

To illustrate this point further we compare the Jackson bounds for best polynomial approximants with actual computations in Chebfun. For example, if f satisfies the Lipschitz condition $|f(x_1) - f(x_2)| \le C|x_1 - x_2|$, $x_1, x_2 \in [-1,1]$, then the approximation error is bounded by [130, Thm. 16.5]

$$\|f - p^*\| \le \frac{C\pi}{2(N+1)}. \tag{6.15}$$

The functions f_5, f_6, f_7 and f_8 in Table 6.1 (among others) are Lipschitz continuous. Using the chebfun command `norm(diff(f),inf)` we calculated the Lipschitz coefficients. Figure 6.10 shows the best approximation errors for N between 1 and 10,000 and the bound (6.15) for f_5 and f_8. The large error of the best approximation to f_8 in Figure 6.10 is consistent with the large Lipschitz constant $C_8 \approx 7 \times 10^2$.

For functions that do not satisfy a Lipschitz condition, one can establish the bound [130, Thm. 16.5]

$$\|f - p^*\| \le \frac{3}{2}\omega_f\Big(\frac{\pi}{N+1}\Big), \tag{6.16}$$

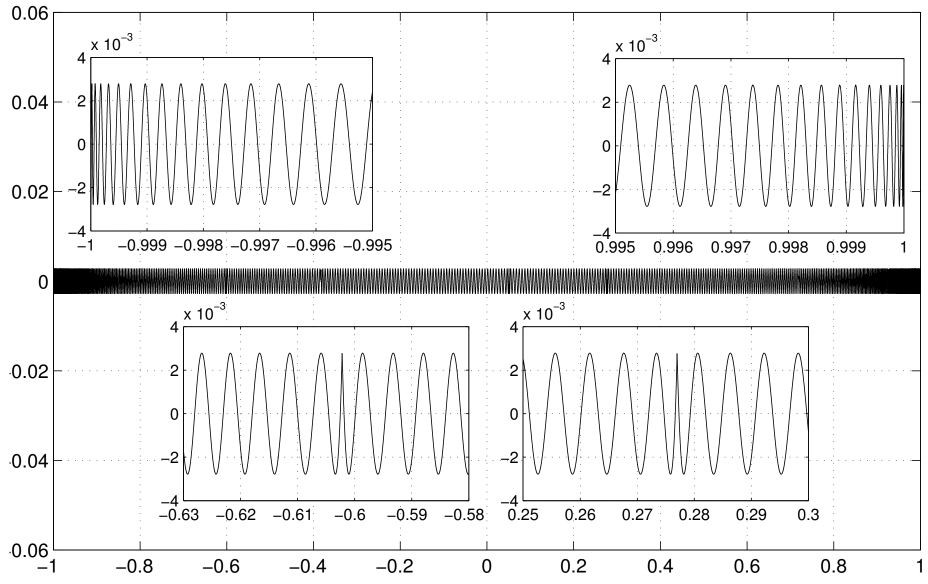

Figure 6.9: Error of the best polynomial approximant of degree 1000 to f_6 in $[-1, 1]$ (cf. Figure 6.7). Details are shown in the surrounding boxes. The computation took 14 seconds.

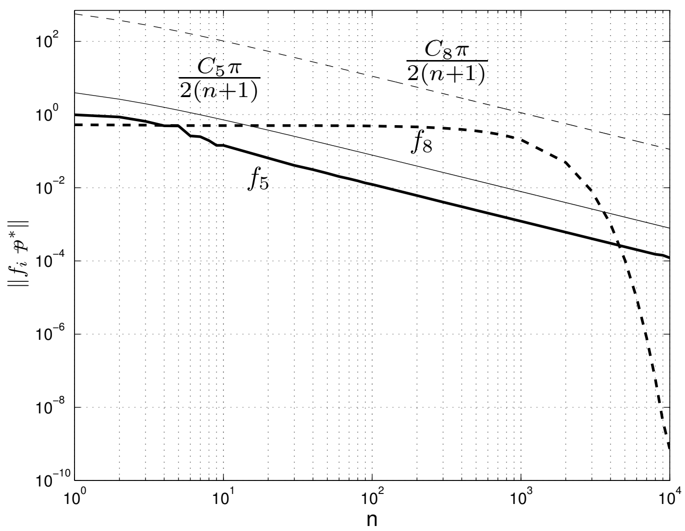

Figure 6.10: Error of best polynomial approximations for functions f_5 and f_8 (bold lines) compared with the Jackson bounds (6.15) for Lipschitz continuous functions (thin lines). The errors for f_6 and f_7, not shown, follow closely the error for f_5.

where ω_f is the modulus of continuity of f (cf. 2.39). For both the functions f_3 and f_4, this bound becomes $\frac{3}{2}\sqrt{\pi/(N+1)}$, and in Figure 6.11 we compare this quantity to the best approximation errors for n from 1 to 1,000.

We mentioned in the introduction that Remez reported in 1934 the coefficients of the best polynomial approximation to $|x|$ in $[-1, 1]$ [136]. Using the overloaded command `poly(p)`,

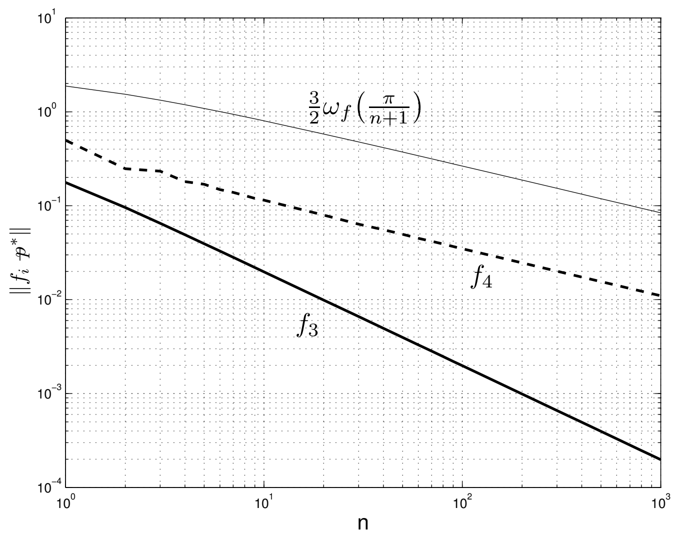

Figure 6.11: Error of best polynomial approximations for functions f_3 and f_4 (bold lines) compared with Jackson bound (6.16) for continuous functions (thin lines). For f_3 the bound is too pessimistic, since (6.16) does not take into account the difference between singularities of f at the endpoints and in the interior.

where `p` is the chebfun of the best polynomial approximation, we can check his numbers. For example, with $N = 11$, it takes about 0.1 seconds to find that the coefficients c_k in the monomial basis $\{x^k\}$ are:

k	c_k (Remez [136])	c_k (Chebfun)
0	0.027837	0.02784511855
2	4.753770	4.75365049278
4	-20.646839	-20.64625015816
6	47.776685	47.77533460523
8	-49.593272	-49.59209097049
10	18.709656	18.70935603064

Evidently Remez's coefficients were accurate to about 4 decimal places.

For this problem of best approximation of $|x|$, much sharper estimates are available than the general bound (6.15). Bernstein [8] proved in 1914 that there exists a positive constant β such that

$$\lim_{N\to\infty} N\||x| - p^*\| = \beta,$$

and from numerical experiments he conjectured that

$$\beta = \frac{1}{2\sqrt{\pi}} = 0.2820947917\ldots.$$

For seventy years this conjecture was open, until Varga and Carpenter [175] proved that it was false and confirmed this with extensive numerical computations. Among their experiments and results, which included sharper lower and upper bounds for β, they computed $N\||x| - p^*\|$ for N up to 104, accurate to nearly 95 decimal places.

The method of Varga and Carpenter requires the use of extended precision. Using the Remez algorithm in Chebfun, we were able to compute the same approximations in 30 seconds on a workstation in ordinary IEEE arithmetic, obtaining the same 52 errors as Varga and Carpenter to between 12 and 15 digits. We also computed the polynomial approximations for degrees up to 1500 and confirmed the first seven of the fifty digits of β that Varga and Carpenter computed by Richardson extrapolation:

$$\beta \approx 0.2801694\ldots$$

We can compute best approximants of higher degrees using the Remez algorithm in Chebfun, and in Figure 6.12 we compare the errors with Bernstein's conjectured number up to $N =$ 10,000, illustrating further — as if further illustration were needed! — that the conjecture was false. However, as we mentioned in item [7] in the Remez code, polynomial interpolation in arbitrary sets of nodes, including the references in the Remez algorithm when these are far from Chebyshev points, may not be well-conditioned. In Figure 6.12, the values for $N > 1500$ are accurate to 4 decimal places.

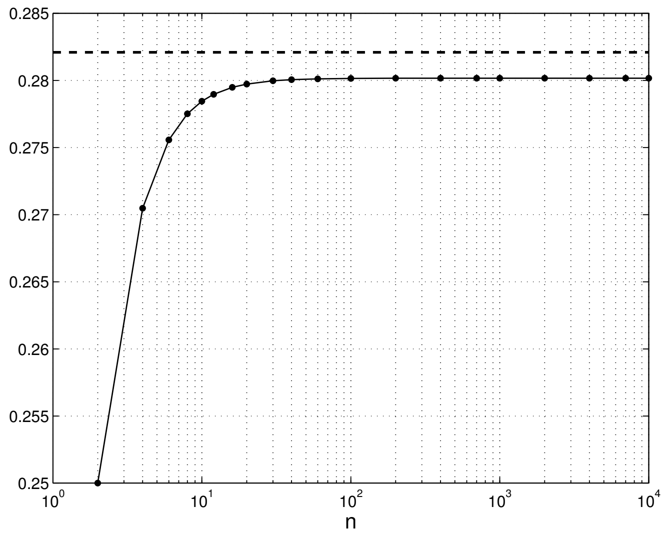

Figure 6.12: Computed values of $N\||x| - p^*\|$ (solid) and Bernstein's conjectured number, $\beta = \frac{1}{2\sqrt{\pi}}$ (dashed) for values of N up to 10,000. As shown by Varga and Carpenter, Bernstein's conjecture was false.

6.5 Implementation for best rational approximation

The code presented above for polynomials can be modified in a few lines for computing best rational approximations. Its range of success, however, is rather different. The robust Remez implementation in the previous sections allows an efficient construction of best polynomials of degrees in the thousands. The rational Remez algorithm, as we will explain, is much more fragile and encounters difficulties even when the degree of the numerator or denominator is small.

The code `rationalremez` in Figure 6.13 has a similar structure to `remez`, and the features [1]–[10] in this code correspond to analogous specifications to the ones explained in Section 6.4 with some notable differences: the levelled error is not obtained from an analytical expression (cf. item [4] in both codes and item [13] in `rationalremez`), from $|h|$ and $q(a_i)$ we compute the values of the numerator of the trial rational (cf. item [6] in both codes), and we construct chebfuns for both the numerator and denominator (cf. item [7] in both codes).

Five additional aspects of the code are marked with the numbers [11]–[15].

11. *Detect degeneracy*: In case that the defect of the rational approximant is nonzero, the Remez algorithm breaks because it assumes that a reference of $N+2$ points can be found. Knowledge of block structures of the *Walsh table*, a similar array to the one for rational interpolants mentioned on p. 67, can be exploited to correct this by modifying the degrees of the numerator and/or denominator of the rational approximant to be computed (see [165] for a discussion of square blocks in various tables of rational approximants). For example, for even functions, the Walsh table is covered with blocks of size 2×2, that is, the best rational approximant is the same for types $[m/n]$, $[m+1/n]$, $[m/n+1]$ and $[m+1/n+1]$, with m and n even. Thus, for a function that is detected to be even and in such a block, the algorithm should automatically set the approximant to one of type $[m/n]$. This is a strategy used in [173] but in the context of CF approximations[2].

12. *Construction of orthogonal matrix*: At each step we need to construct the Vandermonde-type matrix which is orthogonal with respect to the inner product (4.11) which depends on the grid. We form this matrix in one line using the QR factorisation, as we mentioned in Section 4.2. Since our experiments are still limited to references

[2]The following auxiliary code `detectdegeneracy` modifies m and n according to this principle. It was written by Joris van Deun, and is used in the Chebfun `cf` command.

```
function [m,n] = detectdegeneracy(f,m,n), a = chebpoly(chebfun(f,128)); a(end)=2*a(end);
if max(abs(a(end-1:-2:1)))/f.scl < eps,   % f is an even function
   if  (mod(m,2)||mod(n,2)), m = m + 1; elseif mod(m,2) && mod(n,2), n = n - 1; end
elseif max(abs(a(end:-2:1)))/f.scl < eps, % f is an odd function
   if mod(m,2) &&  mod(n,2), m = m + 1; elseif mod(m,2) && mod(n,2), n = n - 1; end
end
```

```
function [p,q,err] = rationalremez(f,m,n)    % compute (m,n) BA to chebfun f

[m,n] = detectdegeneracy(f,m,n); N = m+n;    % modify [m/n] if necessary    [11]
[a,b] = domain(f);                           % endpoints of f
sigma = (-1).^(0:N+1)';                      % alternating signs
normf = norm(f); delta = normf;              % initialise variables
deltamin = delta; iter = 1;                  % initialise variables
[p,q] = chebpade(f,m,n);                     % near-best approximation    [1,15]
[xk,err]=exchange([],0,2,f,p,q,N+2);xo = xk; % initial reference          [1,15]
while (delta/normf > 1e-12) && iter <= 20    % main iteration                [2]
  fk = feval(f,xk);                          % function values
  w = bary_weights(xk);                      % compute barycentric weights   [3]
  [C,~] = qr(fliplr(vander(xk)));            % orthogonal matrix wrt <,>_xk [12]
  ZL=C(:,m+2:N+2).'*diag(fk)*C(:,1:n+1);     % left rational interp matrix  [13]
  ZR=C(:,m+2:N+2).'*diag(sigma)*C(:,1:n+1);  % right rational interp matrix [13]
  [v,d] = eig(ZL,ZR);                        % solve gen eig problem        [13]
  qk_all = C(:,1:n+1)*v;                     % compute all possible qk      [14]
  pos = find(abs(sum(sign(qk_all)) == N+2)); % signs' changes of each qk    [14]
  qk = qk_all(:,pos);                        % keep qk with unchanged sign  [14]
  h = d(pos,pos);                            % levelled reference error      [4]
  if h==0, h = 1e-19; end                    % perturb error if necessary    [5]
  pk = (fk - h*sigma).*qk;                   % vals of r x q in reference    [6]
  p = chebfun(@(x) bary(x,pk,xk,w),N+1);     % chebfun of trial numerator    [7]
  q = chebfun(@(x) bary(x,qk,xk,w),N+1);     % chebfun of trial denominator  [7]
  [xk,err] = exchange(xk,h,2,f,p,q);         % replace reference
  if err/normf > 1e5                         % if overshoot, recompute       [8]
      [xk,err] = exchange(xo,h,1,f,p,q);     %     with one-point exchange
  end
  xo = xk;                                   % copy of xk if overshoot later
  delta = err - abs(h);                      % stopping value                [9]
  if delta < deltamin,                       % store poly with minimal norm [10]
    deltamin = delta;
    pmin = p; qmin = q; errmin = err;
  end
  iter = iter+1;                             % increase counter
end
p = pmin; q = qmin; err = errmin;            % return minimum error and poly
```

Figure 6.13: Code of the Remez algorithm for best rational approximation. The input arguments are a chebfun `f` and the degrees `m` and `n` of the numerator and denominator to be computed and the output arguments are chebfuns `p` and `q` of the best rational approximation to `f` and the error `err`.

with small number of points, we do not require a more efficient implementation, e.g. computing three-term recurrence coefficients.

13. *Generalised eigenvalue problem*: The $(n+1) \times (n+1)$ matrices of system (6.11) are extracted from the orthogonal matrix `C` and the generalised eigenvalue problem is solved with the `eig` command.

14. *Denominator without zeros*: From the $n+1$ eigenvectors of system (6.11) we pick

one in which all entries have the same sign. This is a necessary condition for the denominator to be free of zeros.

15. *Initial reference*: Our last item of discussion coincides with the first one of the polynomial case: an appropriate choice for the initial reference. We start by computing a near-best approximation to f by a Chebyshev–Padé approximation, presented in Section 5.3 and attempt to obtain an initial alternating reference from it[3]. Since the Chebyshev–Padé method is not iterative it is much faster than the Remez computation. Moreover, it has been proposed as a sensible initial guess for the Remez algorithm in [2].

Instead of testing the rational Remez algorithm for different functions, as we did with the polynomial case, we are going to focus on the gamma function $\Gamma(x)$, $x = [0.01, 6]$. To get an appreciation of the range of possibilities of this algorithm, we will compute different configurations of degrees for the numerator and denominator. We begin with four specific cases: [2/2], [4/4], [6/6], and [10/8]. Compare Figure 6.14, where we show the errors of the initial guess, i.e. the error of the Chebyshev–Padé approximant, with Figure 6.15, where we show the errors of the best rational approximants. The supremum errors of the first three cases in Figure 6.15 are approximately 4.634895865905193, 0.02278658329, and 0.000023004075, respectively. For the best approximant of type [10/8], the error is of order 10^{-9}.

In each case the Remez algorithm took 5–6 iterations to obtain the best approximant from the Chebyshev–Padé approximation, and the tolerance was set to 10^{-11}.

The rational Remez algorithm can be used to construct the Walsh table of best rational approximants. There are various theoretical results about these tables, and for the particular case of the gamma function, we refer the reader to [139]. In Figure 6.16 we show the errors in rows and columns of this table, and the first signs of difficulty come into sight. The algorithm does not converge in all cases with the prescribed tolerance, or has to abort due to the presence of poles in a trial function.

To complement the presentation of the Walsh table to the gamma function, we show in the left panel of Figure 6.17 a colour map of the errors for the best rational approximants of type $[m/n]$, $m = 0, \dots, 20$, $n = 0, \dots, 20$. Lighter colours indicate a larger error, but notice that black cells corresponds to entries of the table that could not be computed by our Remez algorithm. Unlike the polynomial case, the Remez algorithm for rational functions has limitations that restrict its use to small values of N, and the computation might break for certain functions. We conclude this section by indicating the following three obstacles that we have observed in our experiments:

[3]We use for this purpose the function `exchange`: when no grid of points is given as a first input argument it returns a grid of points, the size of which is specified by the last input argument, where the error alternates.

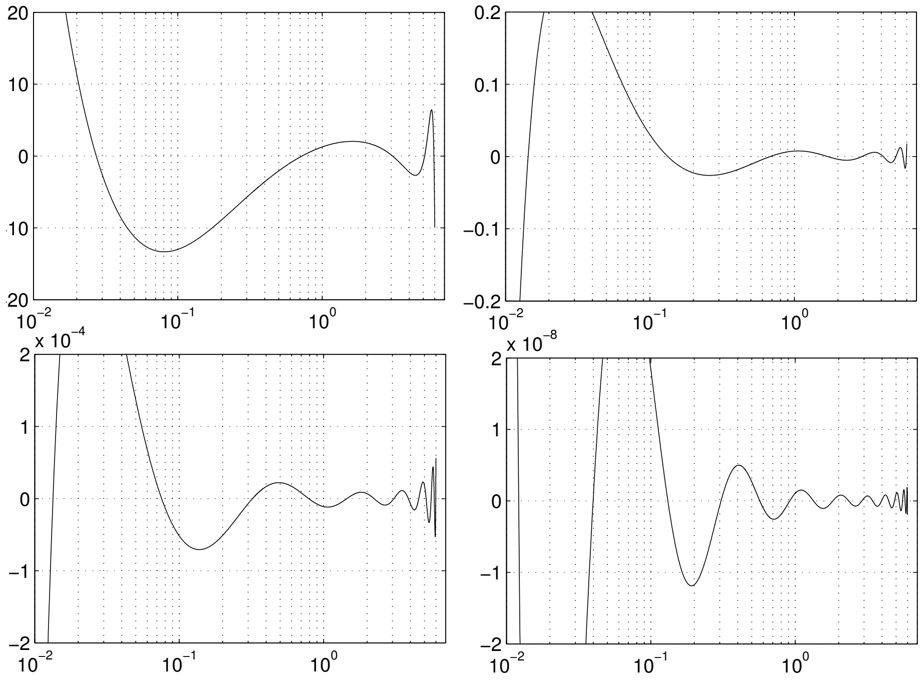
Figure 6.14: Error of Chebyshev–Padé approximants to $\Gamma(x)$, $x = [0.01, 6]$ of types [2/2], [4/4], [6/6], and [10/8] (notice the logarithmic scale on the x-axis). Compare with Figure 6.15.

- *A trial rational function has poles*: In our experiments, the most frequent cause of failure of the algorithm is due to a trial function that has poles. The way in which this happens is when none of the eigenvectors of the system (6.11) corresponds to a zero-free denominator. In that case the algorithm cannot compute a bounded error to locate an appropriate oscillating reference for the next iteration. In theory this situation should not occur if the initial reference is very close to the optimal [98]. Hence, an alternative to this problem would be to use a different initial guess that perhaps is closer to the optimal.

  In the right panel of Figure 6.17 we show the Walsh table when we change the initial guess from Chebyshev-Padé approximation to the rational Carathéodory–Fejér (CF) approximation, implemented in Chebfun with the command `cf` (see [173] and the references therein). The two tables in Figure 6.17 show that indeed the rational Remes is sensitive to the initial approximant, however it is not clear whether CF should always be preferred.

- *The distribution of the optimal reference is ill-conditioned*: Even if the problem with the initial guess is resolved, and if we could find a heuristic way to bypass situations in which all the possible denominators computed from the system (6.11) have zeros, Figure 6.17 suggests that for rational functions whose numerators or denominators

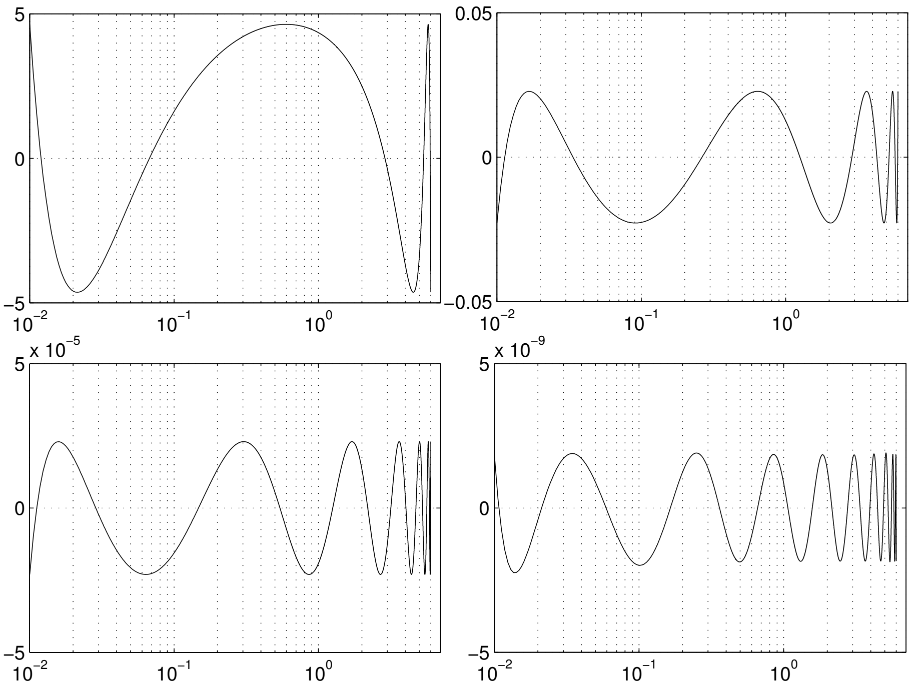

Figure 6.15: Error of best rational approximants to $\Gamma(x)$, $x = [0.01, 6]$ of types [2/2], [4/4], [6/6], and [10/8] (notice the logarithmic scale on the x-axis). The references cluster towards the left where the function has a near-singularity. Compare with Figure 6.14.

are moderately high, the algorithm encounters numerical difficulties.

We have observed that the reason for this failure is due to the lost of accuracy when computing the trial functions and trial references. In particular, the reference in the Remez algorithm tends to cluster towards the singularities of the target function. For $|x|$, for example, the points in the reference approaches the origin very quickly as progressing along the diagonal of the table. As we mentioned in Chapter 5, the evaluation of the rational interpolant with the barycentric formula requires the computation of the polynomial barycentric weights, and for ill-conditioned grids, as the ones formed in the Remez iteration, they vary exponentially.

- *Fast decay of singular values*: In Chapter 5 we saw that in some circumstances, for example when m and n increase simultaneously, the singular values of the rational interpolants decay to machine precision very quickly, introducing artificial factors. Although we are computing a rational interpolant with deviated weights, it is likely that the same phenomenon is also present here. This could be an alternative explanation of the divergence of the Remez algorithm, instead of assuming that the initial guess is far from optimal.

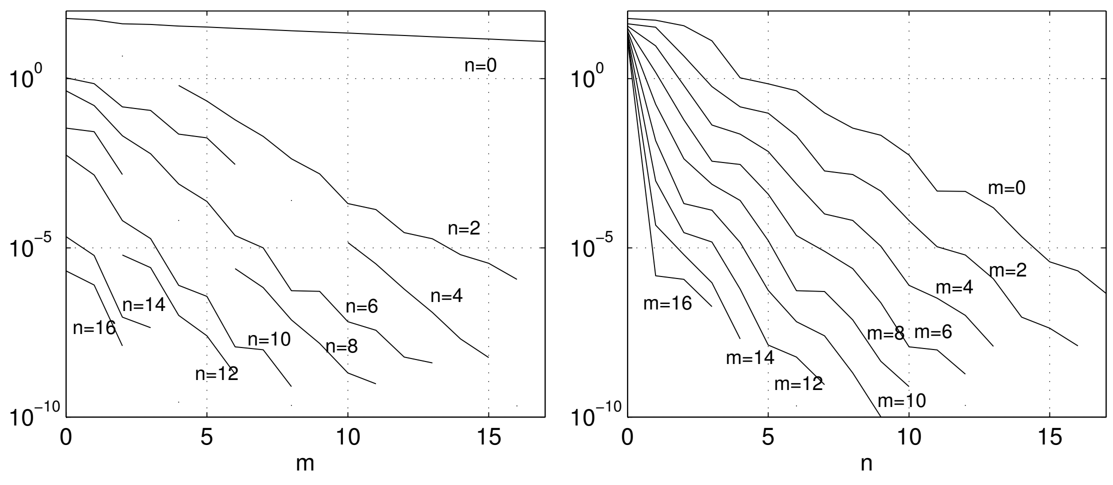

Figure 6.16: Rows and columns of the Walsh table for $\Gamma(x)$, $x = [0.01, 6]$. In the left panel the x-axis represents the numerator degrees and each line shows the behaviour of the error of the best rational approximant for a fixed denominator degree. From upper to lower, these curves indicate the approximants of types $[m/0], [m/2], \ldots, [m/16]$. Notice that not all the curves are complete, signalling that the rational Remez code had to abort for certain pairs (m, n). In particular, notice how the approximants associated with the sixth and eighth rows (third and fifth lines) are discontinuous. Similarly, in the right panel we show the error for columns in the Walsh table. The x-axis corresponds to the degree of the denominator and each line indicates the approximants of types $[0/n], [2/n], \ldots, [16/n]$, from upper to lower. Compare to Figure 6.17.

6.6 Discussion

In this chapter, we have revisited the classic algorithm of Remez for the computation of best approximations with the aid of new ideas and tools developed in this thesis. Our goal has been to obtain robust algorithms that can be used with polynomials or rational functions of very high degrees. For the former, we have accomplished this purpose, and for the latter, we have proposed an algorithm that pursues that vision.

There are two main modifications of the Remez algorithm proposed in this chapter. First, the use of interpolation for the representation of intermediate trial functions. We have derived an analytic expression for the levelled error which simplifies the polynomial problem. For rational functions the expression obtained cannot be used to decouple the problem of interpolation with weighted deviations. We have used the method introduced in Chapter 4 for the computation of rational interpolants, and the derivation of the conditions of the trial function can be formulated as a generalised eigenvalue problem. The second element we bring into the discussion of the Remez algorithm is the use of Chebfun for the computation of the alternating extrema of the error. Accuracy of this computation is required if the best approximants are desired with high precision.

We implemented these ideas in very short algorithms and highlighted specific issues

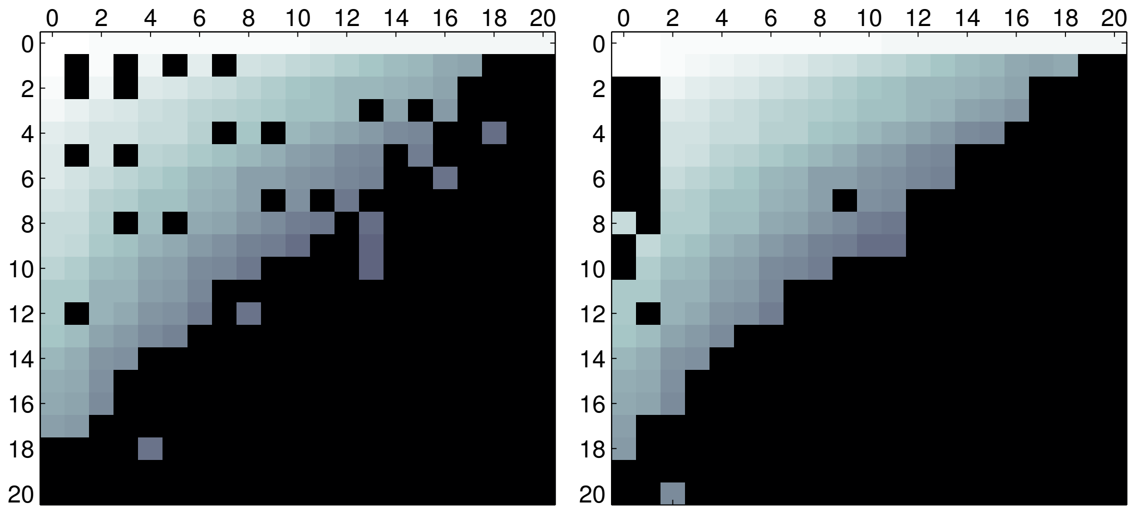

Figure 6.17: Walsh tables for the gamma function $\Gamma(x)$, $x = [0.01, 6]$ using as initial guess Chebyshev–Padé approximation (left) and CF approximation (right). A black box in cell (m, n) indicates that the Remez algorithm failed to compute the best rational approximant of type $[m/n]$. Compare with Figure 6.16.

that are relevant for a robust computation. For polynomials, we have tested the algorithm with degrees in the hundreds and thousands, and we have used it to verify theoretical results of best polynomial approximation. For rational functions we have identified three potential problems that have to be resolved to achieve a completely reliable algorithm: the appearance of poles in a trial approximant, the ill-conditioning of the trial references on which the interpolants are build, and the decay of the singular values of the linear system associated with the interpolation problem which introduces artificial common factors.

CHAPTER 7

Conclusion

Approximation theory lies at the heart of many of the methods used in scientific computing. Interpolation, truncated series and Padé approximation are only a few of the approximation ideas that are commonly incorporated in numerical algorithms. In this thesis we studied and developed algorithms related to some of these topics. Our goal has been the construction of robust algorithms for polynomial and rational approximation, a subject that has been an established part of mathematics for more than a century. We believe that this kind of research demands particular awareness of literature which can be scattered across many decades and countries. In our effort to assess the originality of our own work we have discovered relations with contributions made as long ago as in the time of Cauchy and Jacobi.

The single most important concept in this thesis is that of interpolation. We have discussed its construction, its properties as an approximator, and its application for the construction of other types of approximants. The basis of our research has been the study of polynomial interpolation, and in Chapter 2 we gave a summary of the aspects of it that we consider most relevant.

Our early work was on the improvement of Chebfun for the representation of a wider class of functions than those originally considered, i.e. functions sufficiently smooth to be accurately approximated by polynomial interpolants. Our first contribution was the implementation of the software for constructing and manipulating piecewise-smooth functions in Chebfun version 2. The most visible part of our work on Chebfun in this thesis was the development of the methods to construct the approximants. In Chapter 3 we presented the different ways in which we implement this in the software and in particular we showed the strategy used to automatically introduce breakpoints. The features that make this possible are a recursive subdivision method and an edge-detection mechanism that attempts to locate the singularities from estimates of the derivatives of the function computed from finite differences.

Chebfun has become the laboratory where we test all our ideas and we found it particularly useful for impromptu experimentation, which was important for the development of the algorithms presented in this thesis. Indeed it was the author's success with Remez and other approximation algorithms in Chebfun that launched Nick Trefethen on the project of writing his book, *Approximation Theory and Approximation Practice* [164].

The second part of this thesis corresponded to our work in rational interpolation. For its computation, we have proposed in Chapter 4 a novel algorithm based in the construction of polynomials orthogonal with respect to a certain inner product. The core of this algorithm is the computation of a matrix, denoted Z in this thesis, the kernel of which gives the coefficients of the denominator of the rational interpolant. The matrix Z is obtained from the product of three other matrices: one that maps the coefficients of the denominator to the values at the given reference, one that computes the entrywise product of the denominator and the target function, and one that maps the values back to the coefficients of the numerator of the interpolant. The actual system that is used for the computation is obtained from the appropriate truncation of columns or rows of these matrices.

We have shown that for rational interpolation in roots of unity, this method is particularly simple and can be efficiently implemented with the FFT algorithm. Something similar happens for Chebyshev points of the first kind, and the idea can be slightly modified for the case of Chebyshev points of the second kind. Moreover, we showed how the method can be generalised for arbitrary points. The most similar work that we found to this algorithm was proposed by Eğecioğlu and Koç, but connections with a family of methods initially proposed by Jacobi were also studied.

Having obtained a method for the computation of rational interpolants, we proceeded with the study of its properties for approximation. In Chapter 5 we reviewed some of the results that characterise the behaviour of rational interpolants, and we later compared them with those exhibited by the computed rational interpolants. We found that numerical errors in the computation of rational interpolants can destroy their theoretical properties, and one of the results of this thesis was to characterise the source of this anomaly in terms of the singular values of Z. In particular we found that as more interpolation points are used, the smallest singular values decay very quickly, approaching machine precision. When this happens, the effect caused is the appearance of common factors that do not correspond to those of the true rational interpolant.

The third part in this thesis was the improvement of the classical Remez algorithm for the computation of best polynomial and rational approximations. Our version of the Remez algorithm relies on interpolation and Chebfun. For the polynomial case, the code is extremely robust and we have used it with very high degrees. For the rational case, the algorithm presented here was directly derived from the rational interpolation method introduced in this thesis. The experiments show that the code works well in practice for low degrees of the numerator or denominator, but has difficulties when one attempts to scale it. In our research we have suggested three main reasons why the algorithm may fail, which may give some guidance for its improvement.

Bibliography

[1] M. Almacany, C. Dunham, and J. Williams, *Discrete Chebyshev approximation by interpolating rationals*, IMA J. Numer. Anal., 4 (1984), pp. 467–477. (cited on p. 15)

[2] G. A. Baker and P. R. Graves-Morris, *Padé Approximants*, vol. 59 of Encyclopedia of Mathematics and its Applications, Cambridge University Press, Cambridge, UK, 1996. (cited on pp. 51, 83, 86, 115)

[3] R. W. Barnard, G. Dahlquist, K. Pearce, L. Reichel, and K. C. Richards, *Gram polynomials and the Kummer function*, J. Approx. Theory, 94 (1998), pp. 128–143. (cited on p. 60)

[4] I. Barrodale and C. Phillips, *Solution of an overdetermined system of linear equations in the Chebyshev norm*, ACM Trans. Math. Software, 1 (1975), pp. 264–270. (cited on pp. 95, 108)

[5] Z. Battles, *Numerical Linear Algebra for Continuous Functions*, PhD thesis, University of Oxford, Michaelmas 2005. (cited on p. 39)

[6] Z. Battles and L. N. Trefethen, *An extension of MATLAB to continuous functions and operators*, SIAM J. Sci. Comput., 25 (2004), pp. 1743–1770. (cited on pp. 39, 40, 48)

[7] S. Bernstein, *Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités*, Comm. Soc. Math. Kharkow, 13 (1912/1913), pp. 1–2. (cited on p. 21)

[8] ——, *Sur la meilleure approximation de $|x|$ par des polynômes de degrés donnés*, Acta. Math., 37 (1914), pp. 1–57. (cited on p. 111)

[9] ——, *Quelques remarques sur l'interpolation*, Math. Ann., 79 (1918), pp. 1–12. (cited on p. 22)

[10] J.-P. Berrut, *Rational functions for guaranteed and experimentally well-conditioned global interpolation*, Comput. Math. Appl., 15 (1988), pp. 1–16. (cited on p. 55)

[11] ——, *A matrix for determining lower complexity barycentric representations of rational interpolants*, Numer. Algorithms., 24 (2000), pp. 17–29. (cited on p. 63)

[12] J.-P. Berrut, R. Baltensperger, and H. Mittelmann, *Recent developments in barycentric rational interpolation*, Internat. Ser. Numer. Math., 151 (2005), pp. 27–51. (cited on pp. 55, 63, 67)

[13] J.-P. Berrut and H. Mittelmann, *Matrices for the direct determination of the barycentric weights of rational interpolation*, J. Comput. Appl. Math., 78 (1997), pp. 355–370. (cited on pp. 51, 63, 65, 78, 79, 90)

[14] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Review, 46 (2004), pp. 501–517. (cited on pp. 18, 20, 60)

[15] J. Boothroyd, *Algorithm 318: Chebyschev curve-fit*, Commun. ACM, 10 (1967), pp. 801–803. (cited on p. 93)

[16] E. Borel, *Leçons sur les fonctions de variables réelles*, Gauthier-Villars, Paris, 1905. (cited on p. 95)

[17] F. Bornemann. Private communication to Lloyd N. Trefethen, 2010. (cited on p. 18)

[18] J. A. Boyd, *Computing zeros on a real interval through Chebyshev expansion and polynomial rootfinding*, SIAM J. Numer. Anal., 40 (2002), pp. 1666–1682. (cited on p. 48)

[19] C. Brezinski, *Extrapolation algorithms and Padé approximations: a historical survey*, Appl. Numer. Math., (1996), pp. 299–318. (cited on p. 67)

[20] C. Brezinski and J. van Iseghem, *Padé approximations*, in Handbook of Numerical Analysis, P. G. Ciarlet and J. L. Lions, eds., vol. III, North–Holland, Amsterdam, 1994, pp. 47–222. (cited on p. 83)

[21] ——, *A taste of Padé approximation*, Acta Numerica, (1995), pp. 53–103. (cited on p. 83)

[22] L. Brutman, *Lebesgue functions for polynomial interpolation — a survey*, Ann. Numer. Math., 4 (1997), pp. 111–128. (cited on pp. 33, 34)

[23] ——, *On rational interpolation to $|x|$ at the adjusted Chebyshev nodes*, J. Approx. Theory, 95 (1998), pp. 146–152. (cited on p. 75)

[24] L. Brutman and E. Passow, *Rational interpolation to $|x|$ at the Chebyshev nodes*, Bull. Austral. Math. Soc., 56 (1997), pp. 81–86. (cited on p. 74)

[25] A. Bultheel, *On the block structure of the Laurent-Padé table*, in Rational Approximation and Interpolation, P. R. Graves-Morris, E. B. Saff, and R. S. Varga, eds., vol. 1105 of Lect. Notes in Math., Springer, 1984, pp. 160–169. (cited on p. 86)

[26] A. L. Cauchy, *Sur la formule de Lagrange relative à l'interpolation*, Analyse Algébrique, (1821). (cited on pp. 50, 61)

[27] E. W. Cheney, *Introduction to Approximation Theory*, McGraw-Hill, 1966. (cited on pp. 9, 93, 95, 96)

[28] C. W. Clenshaw and K. Lord, *Rational approximations from Chebyshev series*, in Studies in Numerical Analysis, B. K. P. Scaife, ed., Academic Press, London, 1974, pp. 95–113. (cited on p. 86)

[29] W. J. Cody, W. Fraser, and J. F. Hart, *Rational Chebyshev approximation using linear equations*, Numer. Math., 12 (1968), pp. 242–251. (cited on p. 95)

[30] W. J. Cody and J. Stoer, *Rational Chebyshev approximation using interpolation*, Numer. Math., 9 (1966), pp. 177–188. (cited on p. 95)

[31] A. Córdova, W. Gautschi, and S. Ruscheweyh, *Vandermonde matrices on the circle: spectral properties and conditioning*, Numer. Math., 57 (1990), pp. 577–591. (cited on p. 12)

[32] P. C. Curtis and W. L. Frank, *An algorithm for the determination of the polynomial of best minimax approximation to a function defined on a finite point set*, J. ACM, 6 (1959), pp. 395–404. (cited on p. 108)

[33] G. Dahlquist and Å. Björck, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, N.J., 1974. (cited on p. 60)

[34] P. J. Davis, *Interpolation and Approximation*, Dover Publications, New York, 1975. (cited on pp. 9, 20, 93)

[35] C. de Boor and J. R. Rice, *Extremal polynomials with application to Richardson iteration for indefinite linear systems*, SIAM J. Sci. Stat. Comput, 3 (1982), pp. 47–57. (cited on p. 103)

[36] C. J. de la Vallée Poussin, *Sur les polynômes d'approximation et la représentation approchée d'un angle*, Acad. Roy. de Belg., Bulletins de la Classe des Sci., 12 (1910), pp. 808–844. (cited on p. 96)

[37] R. de Montessus de Ballore, *Sur les fractions continues algébriques*, Bull. Soc. Math. France, 30 (1902), pp. 28–36. (cited on p. 85)

[38] C. B. Dunham, *Choice of basis for Chebyshev approximation*, ACM Trans. Math. Software, 8 (1982), pp. 21–25. (cited on pp. 15, 99)

[39] Ö. Eğecioğlu and Ç. K. Koç, *A fast algorithm for rational interpolation via orthogonal polynomials*, Math. Comp., 53 (1989), pp. 249–264. (cited on pp. 51, 63)

[40] E. Landau, *Abschätzung der Koeffizientsumme einer Potenzreihe*, Archiv der. Math. und Phys., 21 (1913), pp. 250–255. (cited on p. 33)

[41] A. Eisinberg and G. Fedele, *Discrete orthogonal polynomials on equidistant nodes*, International Math. Forum, 21 (2007), pp. 1007–1020. (cited on p. 60)

[42] ——, *Discrete orthogonal polynomials on Gauss-Lobatto Chebyshev nodes*, J. Approx. Theory, 144 (2007), pp. 238–246. (cited on p. 58)

[43] A. Eisinberg, G. Franzè, and N. Salerno, *Asymptotic expansion and estimate of the Landau constant*, Approx. Theory Appl., 17 (2001), pp. 58–64. (cited on p. 33)

[44] P. Erdös, *Problems and results on the theory of interpolation. II*, Acta Math. Hungar., 12 (1961), pp. 235–244. (cited on p. 22)

[45] G. Faber, *Über die interpolatorische Darstellung stetiger Funktionen*, Jahresber. Deutsch. Math. Verein., 23 (1914), pp. 192–210. (cited on p. 22)

[46] R. T. Farouki and V. T. Rajan, *On the numerical condition of polynomials in Bernstein form*, Comput. Aided Geom. Design, 4 (1987), pp. 191–216. (cited on p. 21)

[47] J. Fleischer, *Generalizations of Padé approximants*, in Padé Approximants, P. R. Graves-Morris, ed., Inst. of Phys., Bristol, 1973, pp. 126–131. (cited on p. 86)

[48] M. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numer. Math., 107 (2007), pp. 315–331. (cited on pp. 55, 61)

[49] B. Fornberg, *Numerical differentiation of analytic functions*, ACM Trans. Math. Software, 7 (1981), pp. 512–526. (cited on p. 33)

[50] L. Fox and J. B. Parker, *Chebyshev Polynomials in Numerical Analysis*, Oxford University Press, Oxford, UK, 1968. (cited on p. 13)

[51] A. P. Frankel and W. B. Gragg, *Algorithmic almost best uniform rational approximation with error bounds (abstract)*, SIAM Review, 15 (1973), pp. 418–419. (cited on p. 86)

[52] W. Gautschi, *Norm estimates for inverses of Vandermonde matrices*, Numer. Math., 23 (1975), pp. 337–347. (cited on p. 11)

[53] ———, *Orthogonal polynomials: Applications and computations*, Acta Numerica, 5 (1996), pp. 45–119. (cited on p. 61)

[54] ———, *OPQ: A MATLAB suite of programs for generating orthogonal polynomials and related quadrature rules*. `http://www.cs.purdue.edu/archives/2002/wxg/codes/OPQ.html`, Purdue University, 2004. (cited on p. 61)

[55] ———, *Orthogonal Polynomials: Computation and Approximation*, Oxford University Press, Oxford, 2004. (cited on p. 61)

[56] W. Gautschi and G. Inglese, *Lower bounds for the condition number of Vandermonde matrices*, Numer. Math., 52 (1988), pp. 241–250. (cited on p. 12)

[57] K. O. Geddes, *Near-minimax polynomial approximation in an elliptical region*, SIAM J. Numer. Anal., 15 (1978), pp. 1225–1233. (cited on p. 34)

[58] ———, *Block structure in the Chebyshev-Padé table*, SIAM J. Numer. Anal., 18 (1981), pp. 844–861. (cited on pp. 86, 87)

[59] K. O. Geddes and J. C. Mason, *Polynomial approximation by projections on the unit circle*, SIAM J. Numer. Anal., 12 (1975), pp. 111–120. (cited on p. 33)

[60] L. Gemignani, *Rational interpolation via orthogonal polynomials*, Comput. Math. Appl., 26 (1993), pp. 27–34. (cited on p. 63)

[61] G. H. Golub and L. B. Smith, *Algorithm 414: Chebyshev approximation of continuous functions by a Chebyshev system of functions*, Commun. ACM, 14 (1971), pp. 737–746. (cited on pp. 93, 103, 108)

[62] I. J. Good, *The colleague matrix, a Chebyshev analogue of the companion matrix*, Quart. J. Math., 12 (1961), pp. 61–68. (cited on p. 48)

[63] T. M. M. Graeme J. Byrne and S. J. Smith, *On Lagrange interpolation with equidistant nodes*, Bull. Austral. Math. Soc., 42 (1990), pp. 81–89. (cited on p. 22)

[64] W. B. Gragg, *The Padé table and its relation to certain algorithms of numerical analysis*, SIAM Review, 14 (1972), pp. 1–62. (cited on pp. 67, 83)

[65] ———, *Laurent, Fourier, and Chebyshev-Padé tables*, in Padé and Rational Approximation, E. B. Saff and R. S. Varga, eds., Academic Press, New York, 1977, pp. 61–72. (cited on p. 86)

[66] W. B. Gragg and G. D. Johnson, *The Laurent-Padé table*, in Information Processing 74, North-Holland, 1974, pp. 632–637. (cited on p. 86)

[67] P. R. Graves–Morris, *Symmetrical formulas for rational interpolants*, J. Comput. Appl. Math., 10 (1984), pp. 107–111. (cited on p. 63)

[68] T. H. Gronwall, *A sequence of polynomials connected with the nth roots of unity*, Bull. Amer. Math. Soc., 27 (1921), pp. 275–279. (cited on p. 33)

[69] G. Grünwald, *Über die Divergenzerscheinungen der Lagrangeschen Interpolationspolynome stetiger Funktionen*, Annals of Math., 37 (1936), pp. 908–918. (cited on p. 22)

[70] M. H. Gutknecht, *In what sense is the rational interpolation problem well posed?*, Constr. Approx., 6 (1990), pp. 437–450. (cited on p. 77)

[71] ——, *The rational interpolation problem revisited*, in Proceedings of the U.S.-Western Europe Regional Conference on Padé Approximants and Related Topics (Boulder, CO, 1988), vol. 21, 1991, pp. 263–280. (cited on p. 78)

[72] ——, *Block structure and recursiveness in rational interpolation*, in Approximation theory VII (Austin, TX, 1992), Academic Press, Boston, MA, 1993, pp. 93–130. (cited on p. 78)

[73] ——, *The multipoint Padé table and general recurrences for rational interpolation*, Acta Appl. Math., 33 (1993), pp. 165–194. (cited on p. 78)

[74] S. Güttel, *Rational Krylov Methods for Operator Functions*, PhD thesis, TU Bergakademie Freiberg, 2010. (cited on p. 67)

[75] P. Henrici, *Essentials of Numerical Analysis*, Wiley, New York, 1982. (cited on p. 18)

[76] C. Hermite, *Sur la formule d'interpolation de Lagrange*, J. Reine Angew. Math., 84 (1978), pp. 70–79. (cited on p. 23)

[77] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 2nd. ed., 2002. (cited on pp. 18, 19)

[78] ——, *The numerical stability of barycentric Lagrange interpolation*, IMA J. Numer. Anal., 24 (2004), pp. 547–556. (cited on pp. 18, 19)

[79] IMSL Numerical Libraries, *Technical documentation*, Visual Numerics Inc., Houston, Texas. (cited on p. 93)

[80] C. G. J. Jacobi, *Über die Darstellung einer Reihe gegebner Werthe durch eine gebrochne rationale Function*, J. Reine Angew. Math., 30 (1846), pp. 127–156. (cited on pp. 51, 62)

[81] S. M. Kaber and Y. Maday, *Analysis of some Padé-Chebyshev approximants*, SIAM J. Numer. Anal., 43 (2005), pp. 437–454. (cited on pp. 86, 90)

[82] J. Karlsson and E. Saff, *Singularities of functions determined by the poles of Padé approximants*, in Padé Approximations and its Applications (Amsterdam, 1980), M. G. de Bruin and H. van Rossum, eds., Springer, 1981, pp. 239–254. (cited on p. 74)

[83] E. H. Kaufman, Jr., D. J. Leeming, and G. D. Taylor, *Uniform rational approximation by differential correction and Remes-differential correction*, Internat. J. Numer. Methods Engrg., 17 (1981), pp. 1273–1278. (cited on pp. 93, 95)

[84] P. Kravanja, T. Sakurai, and M. van Barel, *On locating clusters of zeros of analytic functions*, BIT, 39 (1999), pp. 646–682. (cited on p. 74)

[85] L. Kronecker, *Zur Theorie der Elimination einer Variablen aus zwei algebraischen Gleichungen*, Monatsber. Konigl. Preussischen Akad. Wiss. (Berlin), (1881), pp. 535–600. (cited on p. 63)

[86] V. I. Krylov, *Approximate Calculation of Integrals*, Dover, New York, 1962. (cited on pp. 25, 26)

[87] C. Lauter, S. Chevillard, M. Joldes, and N. Jourdan, *User's manual for the Sollya tool*, Arénaire project, ENS de Lyon, Lyon, France. (cited on p. 94)

[88] Le Ba Kkhan' Chin', *Inverse theorems for multipoint Padé approximants*, Math. USSR-Sb., 71 (1992), pp. 149–161. (cited on p. 74)

[89] B. Le Bailly and J. P. Thiran, *Computing complex polynomial Chebyshev approximants on the unit circle by the real Remez algorithm*, SIAM J. Numer. Anal., 36 (1999), pp. 1858–1877. (cited on pp. 93, 108)

[90] K. M. Levasseur, *A probabilistic proof of the Weierstrass approximation theorem*, Amer. Math. Monthly, 91 (1984), pp. 249–250. (cited on p. 21)

[91] A. L. Levin and E. B. Saff, *Potential theoretic tools in polynomial and rational approximation*, in Harmonic Analysis and Rational Approximation: Their Rôles in Signals, Control and Dynamical Systems, J.-D. Fournier, J. Grimm, J. Leblond, and J. R. Partington, eds., vol. 327 of Lecture Notes in Control and Information Sciences, Springer, 2006, pp. 71–94. (cited on p. 25)

[92] X. Li, *On the Lagrange interpolation for a subset of C^k functions*, Constr. Approx., 11 (1995), pp. 287–297. (cited on pp. 28, 29)

[93] G. G. Lorentz, *Approximation of Functions*, Holt, Rinehart and Winston, 1966. (cited on pp. 9, 35, 93, 95)

[94] J. N. Lyness and G. Sande, *Algorithm 413. ENTCAF and ENTCRE: Evaluation of normalized Taylor coefficients of an analytic function*, Comm. ACM, 14 (1971), pp. 669–675. (cited on p. 33)

[95] H. Maehly, *Monthly progress report.* Institute for Advanced Study, Princeton, October 1956. Description of the Electronic Computer Project in `http://library.ias.edu/hs/ecpdofa.html`. (cited on p. 85)

[96] ——, *First interim progress report on rational approximations.* Princeton University, Project NR 044-196, June 23 1958. (cited on p. 85)

[97] ——, *Rational approximations for transcendental functions*, in Proc. of IFIP Congress, 1960. (cited on p. 85)

[98] ——, *Methods for fitting rational approximations,Parts II and III*, J. Assoc. Comp. Mach., 10 (1963), pp. 257–277. (cited on pp. 93, 94, 100, 116)

[99] H. Maehly and C. Witzgall, *Tschebyscheff-Approximationen in kleinen Intervallen II, Stetigkeitssätze für gebrochen rationale Approximationen*, Numer. Math., 2 (1960), pp. 293–307. (cited on p. 77)

[100] Maple, *User Manual*, Waterloo Maple Inc., Waterloo, Canada. (cited on p. 94)

[101] J. Marcinkiewicz, *Sur la divergence des polynômes d'interpolation*, Acta Sci . Math. Szeged, 8 (1937), pp. 131–135. (cited on p. 22)

[102] A. Martínez Finkelshtein, *Acerca de un problema inverso para filas de aproximantes multipuntuales de Padé de tipo newtoniano*, Revista Ciencias Matemáticas, 7 (1986), pp. 43–59. (cited on p. 74)

[103] ——, *On the mth row of Newton type (α, β)-Padé tables and singular points.*, in Approximation and Optimization (Havana, 1987), vol. 1354 of Lecture Notes Math., Springer, 1988, pp. 178–187. (cited on p. 74)

[104] J. Mason and D. Handscomb, *Chebyshev Polynomials*, Chapman and Hall/CRC, Boca Raton, FL, 2003. (cited on pp. 13, 14, 57, 59)

[105] J. C. Mason and A. Crampton, *Laurent-Padé approximants to four kinds of Chebyshev polynomial expansions. Part I. Maehly type approximants*, Numerical Algorithms, 38 (2005), pp. 3–18. (cited on p. 86)

[106] ——, *Laurent-Padé approximants to four kinds of Chebyshev polynomial expansions. Part II. Clenshaw-Lord type approximants*, Numerical Algorithms, 38 (2005), pp. 19–29. (cited on p. 86)

[107] G. Mastroianni and J. Szabados, *Jackson order of approximation by Lagrange interpolation. II*, Acta Math. Hungar., 69 (1995), pp. 73–82. (cited on p. 30)

[108] Mathematica, *The Mathematica GuideBook*, Wolfram Research, Champaign, Illinois. (cited on p. 94)

[109] Matlab, *User Manual*, The MathWorks Inc., Natick, Massachusetts. (cited on p. 94)

[110] J. H. McClellan and T. W. Parks, *A personal history of the Parks-McClellan algorithm*, IEEE Signal Processing Magazine, 22 (2005), pp. 82–86. (cited on p. 108)

[111] J. H. McClellan, T. W. Parks, and L. R. Rabiner, *A computer program for designing optimum FIR linear phase digital filters*, IEEE Trans. Audio Electroacoust., 21 (1973), pp. 506–526. (cited on p. 108)

[112] G. Meinardus, *Approximation of Functions: Theory and Numerical Methods*, Springer, Heidelberg, 1967. (cited on pp. 9, 35, 93, 96)

[113] J. Meinguet, *On the solubility of the Cauchy interpolation problem*, in Approximation Theory, A. Talbot, ed., Academic Press, London, 1970, pp. 137–163. (cited on pp. 51, 61, 63)

[114] C. Méray, *Observations sur la légitimité de l'interpolation*, Annal. Scient. de l'Ecole Normale Supérieure, 3 (1884), pp. 165–176. (cited on p. 22)

[115] ——, *Nouveaux exemples d'interpolations illusoires*, Bull. Sci. Math., 20 (1896), pp. 266–270. (cited on p. 22)

[116] S. N. Mergelyan, *On the representation of functions by series of polynomials on closed sets*, Dokl. Akad. Nauk. SSSR (in Russian), 78 (1951), pp. 405–408. English translation: Amer. Math. Soc. Transl. Ser. 1, vol. 85 (1953). (cited on p. 22)

[117] H. N. Mhaskar and D. V. Pai, *Fundamentals of Approximation Theory*, Narosa Publishing House, New Delhi, 2000. (cited on p. 93)

[118] F. D. Murnaghan and J. J. W. Wrench, *The determination of the Chebyshev approximating polynomial for a differentiable function*, Math. Tabl. Aids Comput., 13 (1959), pp. 185–193. (cited on pp. 93, 105, 108)

[119] NAG, *Library Manual*, The Numerical Algorithms Group Ltd., Oxford, UK. (cited on p. 94)

[120] D. J. Newman, *Rational approximation to $|x|$*, Michigan Math. J., 11 (1964), pp. 11–14. (cited on p. 75)

[121] E. M. Nikishin and V. N. Sorokin, *Rational Approximations and Orthogonality*, vol. 92 of Translation of Mathematical Monographs, American Mathematical Society, Providence, Rhode Island, 1991. Translated from the Russian by Ralph P. Boas. (cited on p. 67)

[122] G. Opitz, *Steigungsmatrizen*, Z. Angew. Math. Mech., 44 (1964), pp. T52–T54. (cited on p. 63)

[123] R. Pachón, P. Gonnet, and J. van Deun, *Fast and stable rational interpolation in roots of unity and Chebyshev points*, (2010). SIAM J. Numer. Anal., submitted (available at Oxford University Mathematical Institute, Numerical Analysis Group Report, No. 10/02). (cited on pp. 50, 66)

[124] R. Pachón, R. Platte, and L. N. Trefethen, *Piecewise smooth chebfuns.* IMA J. Numer. Anal., (in press, published online on July 2009). (cited on pp. 37, 40, 46, 47)

[125] 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. (cited on pp. 92, 99)

[126] T. W. Parks and J. H. McClellan, *Chebyshev approximation for nonrecursive digital filters with linear phase*, IEEE Trans. Circuit Theory, 19 (1972), pp. 189–194. (cited on pp. 93, 94, 99)

[127] A. Pinkus, *Weierstrass and approximation theory*, J. Approx. Th., 107 (2000), pp. 1–66. (cited on p. 21)

[128] R. B. Platte, L. N. Trefethen, and A. B. J. Kuijlaars, *Impossibility of fast stable approximation of analytic functions from equispaced samples.* SIAM Review (to appear). (cited on p. 61)

[129] M. Polezzi and A. Sri Ranga, *On the denominator values and barycentric weights of rational interpolants*, J. Comput. Appl. Math., 200 (2007), pp. 576–590. (cited on p. 63)

[130] M. J. D. Powell, *Approximation Theory and Methods*, Cambridge University Press, Cambridge, UK, 1981. (cited on pp. 9, 93, 95, 96, 98, 101, 103, 109)

[131] A. Predonzan, *Su una formula d'interpolazione per le funzioni razionali*, Rend. Sem. Mat. Univ. Padova, 22 (1953), pp. 417–425. (cited on p. 63)

[132] P. Rabinowitz, *Applications of linear programming to numerical analysis*, SIAM Review, 10 (1968), pp. 121–159. (cited on p. 95)

[133] H.-J. Rack and M. Reimer, *The numerical stability of evaluation schemes for polynomials based on the Lagrange interpolation form*, BIT, 22 (1982), pp. 101–107. (cited on p. 18)

[134] L. Reichel and G. Opfer, *Chebyshev–Vandermonde systems*, Math. Comp., 57 (1991), pp. 703–721. (cited on p. 14)

[135] E. Y. Remez, *Sur la détermination des polynômes d'approximation de degré donné*, Comm. Soc. Math. Kharkov, 10 (1934). (cited on p. 93)

[136] ——, *Sur le calcul effectif des polynomes d'approximation de Tchebychef*, Compt. Rend. Acad. Sci., 199 (1934), pp. 337–340. (cited on pp. 93, 102, 108, 110, 111)

[137] ——, *Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation*, Compt. Rend. Acad. Sci., 198 (1934), pp. 2063–2065. (cited on p. 93)

[138] J. R. Rice, *The Approximation of Functions*, vol. 1, Addison-Wesley, 1964. (cited on pp. 9, 93)

[139] ——, *On the L_∞ Walsh arrays for* $\Gamma(x)$ *and* Erfc(x), Math. Comp., 18 (1964), pp. 617–626. (cited on p. 115)

[140] T. J. Rivlin, *The Chebyshev Polynomials*, John Wiley, New York, 1974. (cited on pp. 13, 59)

[141] W. Rudin, *Real and Complex Analysis*, McGraw-Hill International Editions, Singapore, 3rd ed., 1987. (cited on pp. 21, 22, 66)

[142] C. Runge, *Zur Theorie der eindeutigen analytischen Funktionen*, Acta Math., 6 (1885), pp. 229–244. (cited on p. 21)

[143] ——, *Über empirische Funktionen und die Interpolation zwischen äquidistanten Ordinaten*, Z. Math. Phys., 46 (1901), pp. 224–243. (cited on p. 22)

[144] E. B. Saff, *An extension of Montessus de Ballore's theorem on the convergence of interpolating rational functions*, J. Approx. Theory, 6 (1972), pp. 63–67. (cited on pp. 69, 71)

[145] ——, *An introduction to the convergence theory of Padé approximants*, in Aspects of Contemporary Complex Analysis, D. A. Brannan and J. G. Clunie, eds., New York, 1980, Academic Press, pp. 493–502. (cited on p. 69)

[146] ——, *Polynomial and rational approximation in the complex domain*, in Approximation Theory, C. de Boor, ed., vol. 36 of Proc. Symp. Appl. Math., Providence, 1986, Amer. Math. Soc, pp. 21–49. (cited on p. 83)

[147] E. B. Saff and J. L. Walsh, *On the convergence of rational functions which interpolate in the roots of unity*, Pacific J. Math., 45 (1973), pp. 639–641. (cited on pp. 56, 71)

[148] O. Salazar Celis, *Practical Rational Interpolation of Exact and Inexact Data. Theory and Algorithms*, PhD thesis, University of Antwerp, July 2008. (cited on pp. 54, 66)

[149] H. E. Salzer, *Rational interpolation using incomplete barycentric forms*, Z. Angew. Math. Mech., 61 (1981), pp. 161–164. (cited on pp. 61, 63)

[150] F. W. Sauer, *Algorithm 604: A FORTRAN program for the calculation of an extremal polynomial*, ACM Trans. Math. Software, 9 (1983), pp. 381–383. (cited on p. 93)

[151] H. Schmitt, *Algorithm 409, discrete Chebychev curve fit*, Comm. ACM, 14 (1971), pp. 355–356. (cited on p. 108)

[152] C. Schneider and W. Werner, *Some new aspects of rational interpolation*, Math. Comp., 47 (1986), pp. 285–299. (cited on pp. 51, 55, 63)

[153] A. Shenitzer, *Chebyshev approximation of a continuous function by a class of functions*, J. Assoc. Comp. Mach., 4 (1957), pp. 30–35. (cited on p. 93)

[154] J. C. Simpson, *Fortran translation of algorithm 409, Discrete Chebychev curve fit*, ACM Trans. Math. Software, 2 (1976), pp. 95–97. (cited on pp. 95, 108)

[155] K. Singhal and J. Vlach, *Accuracy and speed of real and complex interpolation*, Computing, 11 (1973), pp. 147–158. (cited on p. 12)

[156] W. Specht, *Die Lage der Nullstellen eines Polynoms. IV*, Math. Nachr., 21 (1960), pp. 201–222. (cited on p. 48)

[157] H. Stahl, *Convergence of rational interpolants*, in Numerical Analysis—A numerical analysis conference in honour of Jean Meinguet, Bull. Belg. Math. Soc. Simon Stevin, 1996, pp. 11–32. Suppl. vol. 3. (cited on p. 90)

[158] K.-G. Steffens, *The History of Approximation Theory. From Euler to Bernstein*, Birkhäuser, Boston, 2006. (cited on p. 93)

[159] E. L. Stiefel, *Numerical methods of Tchebycheff approximation*, in On Numerical Approximation, R. Langer, ed., University of Wisconsin Press, Madison, 1959, pp. 217–232. (cited on p. 108)

[160] G. Szegö, *Orthogonal polynomials*, vol. 23 of Colloquium Publications, Amer. Math. Soc., Providence, R.I., 4th ed., 1975. (cited on p. 30)

[161] R. Taylor and V. Totik, *Lebesgue constants for Leja points*, IMA J. Numer. Anal., to appear. (cited on p. 19)

[162] T. W. Tee and L. N. Trefethen, *A rational spectral collocation method with adaptively transformed Chebyshev grid points*, SIAM J. Sci. Comput., 28 (2006), pp. 1798–1811. (cited on pp. 67, 86)

[163] F. B. Tobin A. Driscoll and L. N. Trefethen, *The chebop system for automatic solution of differential equations*, BIT, 48 (2008), pp. 701–723. (cited on p. 39)

[164] L. N. Trefethen, *Approximation Theory and Approximation Practice: A 21st-Century Treatment in the Form of 32 Executable Chebfun M-files.* Book in preparation. (cited on pp. 9, 30, 33, 34, 49, 121)

[165] ——, *Square blocks and equioscillation in the Padé, Walsh, and CF tables*, in Rational Approximation and Interpolation, P. Graves-Morris, E. Saff, and R. Varga, eds., vol. 1105 of Lect. Notes in Math., Springer, 1984. (cited on pp. 96, 113)

[166] ——, *Computing numerically with functions instead of numbers*, Math. in Comp. Sci., 1 (2007), pp. 9–19. (cited on p. 38)

[167] ——, *Is Gauss quadrature better than Clenshaw–Curtis?*, SIAM Review., 50 (2008), pp. 67–87. (cited on p. 67)

[168] L. N. Trefethen and D. Bau III, *Numerical Linear Algebra*, SIAM, Philadelphia, PA., 1997. (cited on pp. 11, 54, 61)

[169] L. N. Trefethen and M. H. Gutknecht, *Padé, stable Padé, and Chebyshev-Padé approximation*, in Algorithms for Approximation, J. Mason, ed., Clarendon Press, 1987. (cited on p. 88)

[170] L. N. Trefethen, N. Hale, R. B. Platte, T. A. Driscoll, and R. Pachón, *Chebfun version 3.* `http://www.maths.ox.ac.uk/chebfun/`, Oxford University, 2009. (cited on p. 39)

[171] L. N. Trefethen and J. Weideman, *Two results on polynomial interpolation in equally spaced points*, J. Approx. Theory, 65 (1991), pp. 247–260. (cited on pp. 33, 34)

[172] W. van Assche, *Padé and Hermite-Padé approximation and orthogonality*, Surveys in Approximation Theory, 2 (2006), pp. 61–91. Online article at `http://www.math.technion.ac.il/sat`. (cited on p. 83)

[173] J. van Deun and L. N. Trefethen, *A robust implementation of the Carathéodory–Fejér method.* Submitted. (cited on pp. 113, 116)

[174] C. van Loan, *Computational Frameworks for the Fast Fourier Transform*, SIAM, Philadelphia, 1992. (cited on p. 12)

[175] R. S. Varga and A. J. Carpenter, *On the Bernstein conjecture in approximation theory*, Constr. Approx, 1 (1985), pp. 333–348. (cited on p. 112)

[176] L. Veidinger, *On the numerical determination of the best approximation in the Chebyshev sense*, Numer. Math., 2 (1960), pp. 99–105. (cited on p. 103)

[177] J. L. Walsh, *Interpolation and Approximation by Rational Functions in the Complex Domain*, vol. 20 of Colloquium Publications, American Mathematical Society, Providence, Rhode Island, 5th ed., 1969. (cited on pp. 9, 90, 93)

[178] J. A. C. Weideman and L. N. Trefethen, *The eigenvalues of second-order spectral differentiation matrices*, SIAM J. Numer. Anal., 25 (1988), pp. 1279–1298. (cited on p. 31)

[179] K. WEIERSTRASS, *Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen*, Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin, (1885), pp. 633–639 (first part) and 789–805 (second part). (cited on p. 21)

[180] H. WERNER, *Rationale Tshcebyscheff-Approximation, Eigenwerttheorie und Differenzenrechnung*, Arch. Rat. Mech. Anal., 13 (1963), pp. 330–347. (cited on p. 63)

[181] ——, *Rationale Interpolation von* $|x|$ *in äquidistanten Punkten*, Math. Z., 180 (1982), pp. 11–17. (cited on p. 74)

[182] L. WUYTACK, *On some aspects of the rational interpolation problem*, SIAM J. Numer. Anal., 11 (1974), pp. 52–60. (cited on pp. 51, 77, 78)

[183] T. F. XIE AND S. P. ZHOU, *The asymptotic property of approximation to* $|x|$ *by Newman's rational operators*, Acta Math. Hungar., 103 (2004), pp. 313–319. (cited on p. 75)

[184] L. C. YOUNG, *Limits of Stieltjes integrals*, J. London Math. Soc., s1-9 (1934), pp. 119–126. (cited on p. 26)

[185] S. P. ZHOU AND L. Y. ZHU, *Convergence of Lagrange interpolation polynomials for piecewise smooth functions*, Acta Math. Hungar., 93 (2001), pp. 71–76. (cited on p. 30)

[186] X. ZHU AND G. ZHU, *A method for directly finding the denominator values of rational interpolants*, J. Comput. Appl. Math., 148 (2002), pp. 341–348. (cited on p. 63)