Approximation Theory and Approximation Practice

Lloyd N. Trefethen

With 163 figures and 210 exercises.

For Kate

Contents

1. Introduction

Welcome to a beautiful subject!—the constructive approximation of functions. And welcome to a rather unusual book.

Approximation theory is an established field, and my aim is to teach you some of its most important ideas and results, centered on classical topics related to polynomials and rational functions. The style of this book, however, is quite different from what you will find elsewhere. Everything is illustrated computationally with the help of the Chebfun software package in Matlab, from Chebyshev interpolants to Lebesgue constants, from the Weierstrass approximation theorem to the Remez algorithm. Everything is practical and fast, so we will routinely compute polynomial interpolants or Gauss quadrature weights for tens of thousands of points. In fact, each chapter of this book is a single Matlab M-file, and the book has been produced by executing these files with the Matlab "publish" facility. The chapters come from M-files called `chap1.m`, ..., `chap28.m` and you can download them and use them as templates to be modified for explorations of your own.

Beginners are welcome, and so are experts, who will find familiar topics approached from new angles and familiar conclusions turned on their heads. Indeed, the field of approximation theory came of age in an era of polynomials of degrees perhaps O(10). Now that O(1000) is easy and O(1,000,000) is not hard, different questions come to the fore. For example, we shall see that "best" approximants are hardly better than "near-best", though they are much harder to compute, and that, contrary to widespread misconceptions, numerical methods based on high-order polynomials can be extremely efficient and robust.

This is a book about approximation, not Chebfun, and for the most part we

shall use Chebfun tools with little explanation. For information about Chebfun, see `http://www.maths.ox.ac.uk/chebfun`. In the course of the book we shall use Chebfun overloads of the following Matlab functions, among others:

```
CONV, CUMSUM, DIFF, INTERP1, NORM, POLY, POLYFIT, ROOTS, SPLINE
```

as well as additional Chebfun commands such as

```
CF, CHEBELLIPSEPLOT, CHEBPADE, CHEBPOLY, CHEBPTS, LEBESGUE,
LEGPOLY, LEGPTS, PADEAPPROX, RATINTERP, REMEZ.
```

There are quite a number of excellent books on approximation theory. Three classics are [Cheney 1966], [Davis 1975], and [Meinardus 1967], and a slightly more recent computationally oriented classic is [Powell 1981]. Perhaps the first approximation theory text was [Borel 1905].

A good deal of my emphasis will be on ideas related to Chebyshev points and polynomials, whose origins go back more than a century to mathematicians including Chebyshev (1821–1894), de la Vallée Poussin (1866–1962), Bernstein (1880–1968), and Jackson (1888–1946). In the computer era, some of the early figures who developed "Chebyshev technology," in approximately chronological order, were Lanczos, Clenshaw, Babenko, Good, Fox, Elliott, Mason, Orszag, and V. I. Lebedev. Books on Chebyshev polynomials have been published by Snyder [1966], Fox and Parker [1968], Paszkowski [1975], Rivlin [1990], and Mason and Handscomb [2003]. One reason we emphasize Chebyshev technology so much is that in practice, for working with functions on intervals, these methods are unbeatable. For example, we shall see in Chapter 16 that the difference in approximation power between Chebyshev and "optimal" interpolation points is utterly negligible. Another reason is that if you know the Chebyshev material well, this is the best possible foundation for work on other approximation topics, and for understanding the links with Fourier analysis.

My style is conversational, but that doesn't mean the material is all elementary. The book aims to be more readable than most, and the numerical experiments help achieve this. At the same time, theorems are stated and proofs are given, often rather tersely, without all the details spelled out. It is assumed that the reader is comfortable with rigorous mathematical arguments and familiar with ideas like continuous functions on compact sets, Lipschitz continuity, contour integrals in the complex plane, and norms of operators. If you are a student, I hope you are an advanced undergraduate or graduate who has taken courses in numerical analysis and complex analysis. If you are a seasoned mathematician, I hope you are also a Matlab user.

Each chapter has a collection of exercises, which span a wide range from mathematical theory to Chebfun-based numerical experimentation. Please do not skip the numerical exercises! If you are going to do that, you might as well put this book aside and read one of the classics from the 1960s.

To give readers easy access to all the examples in executable form, the book was produced using `publish` in LaTeX mode: thus this chapter, for example, can be generated with the Matlab command `publish('chap1','latex')`. To

achieve the desired layout, we begin each chapter by setting a few default parameters concerning line widths for plots, etc., which are collected in an M-file called `ATAPformats` that is included with the standard distribution of Chebfun. Most readers can ignore these details and simply apply `publish` to each chapter. For the actual production of the printed book, `publish` was executed not chapter-by-chapter but on a concatenation of all the chapters, and a few tweaks were made to the resulting LaTeX file, including removal of Matlab commands whose effects are evident from looking at the figures, like `title`, `axis`, `hold off`, and `grid on`.

The Lagrange interpolation formula was discovered by Waring, the Gibbs phenomenon was discovered by Wilbraham, and the Hermite integral formula is due to Cauchy. These are just some of the instances of Stigler's Law in approximation theory, and in writing this book I have taken pleasure in trying to cite the originator of each of the main ideas. Thus the entries in the References stretch back several centuries, and each has an editorial comment attached. Often the original papers are surprisingly readable and insightful, at least if you are comfortable with French or German, and in any case, it seems particularly important to pay heed to original sources in a book like this that aims to reexamine material that has grown too standardized in the textbooks. Another reason for looking at original sources is that in the last few years it has become far easier to track them down, thanks to the digitization of journals, though there are always difficult special cases like [Wilbraham 1848], which I finally found in an elegant leather-bound volume in the Balliol College library. No doubt I have missed originators of certain ideas, and I would be glad to be corrected on such points by readers. For a great deal of information about approximation theory, including links to dozens of classic papers, see the History of Approximation Theory web site at `http://www.math.technion.ac.il/hat/`.

Perhaps I may add a further personal comment. As an undergraduate and graduate student in the late 1970s and early 1980s, one of my main interests was approximation theory. I regarded this subject as the foundation of my wider field of numerical analysis, but as the years passed, research in approximation theory came to seem to me dry and academic, and I moved into other areas. Now times have changed, computers have changed, and my perceptions have changed. I now again regard approximation theory as exceedingly close to computing, and in this book we shall discuss many practical numerical problems, including interpolation, quadrature, rootfinding, analytic continuation, extrapolation of sequences and series, and solution of differential equations.

Why is approximation theory useful? The answer goes much further than the rather tired old fact that your computer relies on approximations to evaluate functions like $\sin(x)$ and $\exp(x)$. For my personal answer to the question, concerning polynomials and rational functions in particular, take a look at the last three pages of Chapter 23, beginning with the quotes of Runge and Kirchberger from the beginning of the 20th century. There are also many other fascinating and important topics of approximation theory not touched upon in this volume, including splines, wavelets, radial basis functions, compressed sensing, and multivariate approximations of all kinds.

In summary, here are some distinctive features of this book:

- The emphasis is on topics close to numerical algorithms.
- Everything is illustrated with Chebfun.
- Each chapter is a publishable M-file, available online.
- There is a bias toward theorems and methods for analytic functions, which appear so often in applications, rather than on functions at the edge of discontinuity with their seductive theoretical challenges.
- Original sources are cited rather than textbooks, and each item in the bibliography is listed with an editorial comment.

At a more detailed level, virtually every chapter contains mathematical and scholarly novelties. Examples are the use of barycentric formulas beginning in Chapter 5, the tracing of barycentric formulas and the Hermite integral formula back to Jacobi in 1825 and Cauchy in 1826, Theorem 7.1 on the size of Chebyshev coefficients, the introduction to potential theory in Chapter 12, the discussion in Chapter 14 of prevailing misconceptions about interpolation, the presentation of colleague matrices for rootfinding in Chapter 18 with Jacobi matrices for quadrature as a special case in Chapter 19, Theorem 19.5 showing that Clenshaw–Curtis quadrature converges about as fast as Gauss quadrature, the first textbook presentation of Carathódory–Fejér approximation in Chapter 20, the explanation in Chapter 22 of why polynomials are not optimal functions for linear approximation, the extensive discussion in Chapter 23 of the uses of rational approximations, and the SVD-based algorithms for robust rational interpolation and linearized least-squares fitting and Padé approximation in Chapters 26 and 27.

All in all, we shall see that there is scarcely an idea in classical approximation theory that cannot be illustrated in a few lines of Chebfun code, and as I first imagined around 1975, anyone who wants to be expert at numerical computation really does need to know this material.

Dozens of people have helped me in preparing this book. I cannot name them all, but I would like to thank in particular Serkan Gugercin, Nick Higham, Jörg Liesen, Ricardo Pachón, and Ivo Panayotov for reading the whole text and making many useful suggestions, Jean-Paul Berrut for teaching me about rational functions and barycentric formulas, Folkmar Bornemann for bringing to light historical surprises involving Jacobi, Cauchy, and Marcel Riesz, and Volker Mehrmann for hosting a sabbatical visit to the Technical University of Berlin in 2010 during which much of the work was done. I am grateful to Max Jensen of the University of Durham, whose invitation to give a 50-minute talk in March 2009 sparked the whole project, and to Marlis Hochbruck and Caroline Lasser for testing a draft of the book with their students in Karlsruhe and Munich. Here in the Numerical Analysis Group at Oxford, Endre Süli and Andy Wathen have been the finest colleagues one could ask for these past fifteen years, and the remarkable Lotti Ekert makes everything run smoothly. Finally, none of this

would have been possible without the team who have made Chebfun so powerful and beautiful, my good friends Zachary Battles, Ásgeir Birkisson, Toby Driscoll, Pedro Gonnet, Stefan Güttel, Nick Hale, Ricardo Pachón, Rodrigo Platte, Mark Richardson, and Alex Townsend.

Exercise 1.1. Chebfun download. Download Chebfun from the web site at `http://www.maths.ox.ac.uk/chebfun` and install it in your Matlab path as instructed there. Execute `chebtest` to make sure things are working, and note the time taken. Execute `chebtest` again and note how much speedup there is now that various files have been brought into memory. Now read Chapter 1 of the online *Chebfun Guide,* and look at the list of Examples.

Exercise 1.2. The publish command. Execute `help publish` and `doc publish` in Matlab to learn the basics of how the `publish` command works. Then download the files `chap1.m` and `chap2.m` from `http://www.maths.ox.ac.uk/chebfun/ATAP` and publish them with `publish('chap1','latex')` followed by appropriate LaTeX commands. If you are a student taking a course for which you are expected to turn in writeups of the exercises, I recommend that you make it your habit to produce them with `publish`.

Exercise 1.3. Textbook X. Buy or borrow a copy of an approximation theory textbook, which we shall call X; good examples are the books of Achieser, Braess, Cheney, Davis, Lorentz, Meinardus, Natanson, Powell, Rice, Rivlin, Schönhage, Timan, and Watson listed in the References. As you work through *Approximation Theory and Approximation Practice*, keep X at your side and get in the habit of comparing treatments of each topic between *ATAP* and X. (a) What are the author, title, and publication date of X? (b) Where did/does the author work and what were/are his/her dates? (c) Look at the first three theorems in X and write down one of them that interests you. You do not have to write down the proof.

2. Chebyshev points and interpolants

Any interval $[a, b]$ can be scaled to $[-1, 1]$, so most of the time, we shall just talk about $[-1, 1]$.

Let n be a positive integer:

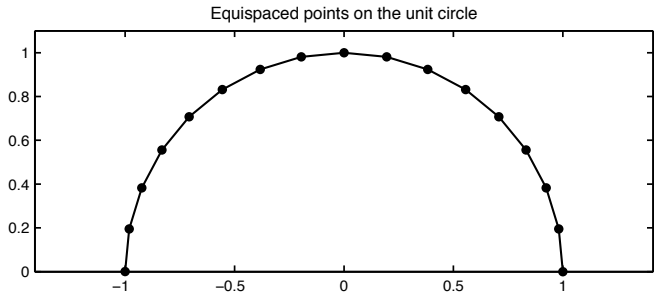
```
n = 16;
```

Consider $n + 1$ equally spaced angles $\{\theta_j\}$ from 0 to π:

```
tt = linspace(0,pi,n+1);
```

We can think of these as the arguments of $n+1$ points $\{z_j\}$ on the upper half of the unit circle in the complex plane. These are the $(2n)$th roots of unity lying in the closed upper half-plane:

```
zz = exp(1i*tt); plot(zz,'.-k')
```



The *Chebyshev points* associated with the parameter n are the real parts of these points,

$$x_j = \mathrm{Re}\, z_j = \frac{1}{2}(z_j + z_j^{-1}), \quad 0 \le j \le n : \tag{2.1}$$

```
xx = real(zz);
```

Some authors use the terms *Chebyshev–Lobatto points*, *Chebyshev extreme points*, or *Chebyshev points of the second kind*, but as these are the points most often used in practical computation, we shall just say Chebyshev points.

Another way to define the Chebyshev points is in terms of the original angles,

$$x_j = \cos(j\pi/n), \quad 0 \le j \le n, \tag{2.2}$$

```
xx = cos(tt);
```

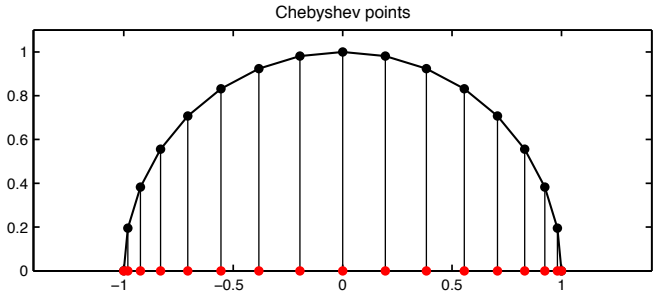
and the problem of polynomial interpolation in these points was considered at least as early as [Jackson 1913]. There is also an equivalent Chebfun command `chebpts`:

```
xx = chebpts(n+1);
```

Actually this result isn't exactly equivalent, as the ordering is left-to-right rather than right-to-left. Concerning rounding errors when these numbers are calculated numerically, see Exercise 2.3.

Let us add the Chebyshev points to the plot:

```
hold on, for j = 2:n, plot([xx(n+2-j) zz(j)],'k'), end
plot(xx,0*xx,'.r')
```



They cluster near 1 and -1, with the average spacing as $n \to \infty$ being given by a density function with square root singularities at both ends (Exercise 2.2).

Let $\{f_j\}$, $0 \le j \le n$, be a set of numbers, which may or may not come from sampling a function $f(x)$ at the Chebyshev points. Then there exists a unique polynomial p of degree n that interpolates these data, i.e., $p(x_j) = f_j$ for each j.

When we say "of degree n," we mean of degree less than or equal to n, and we let $\mathcal{P}_n$ denote the set of all such polynomials:

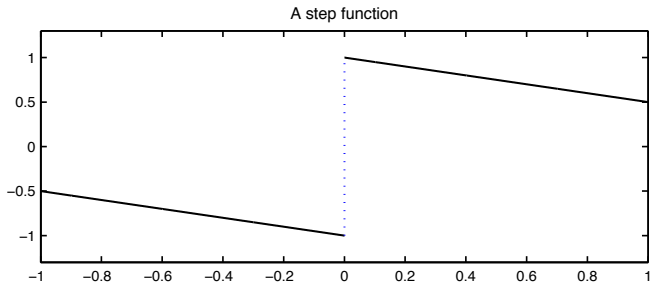
$$\mathcal{P}_n = \{\text{polynomials of degree at most } n\}. \tag{2.3}$$

As we trust the reader already knows, the existence and uniqueness of polynomial interpolants applies for any distinct set of interpolation points. In the case of Chebyshev points, we call the polynomial the *Chebyshev interpolant.*

Polynomial interpolants through equally spaced points have terrible properties, as we shall see in Chapters 11–15. Polynomial interpolants through Chebyshev points, however, are excellent. It is the clustering near the ends of the interval that makes the difference, and other sets of points with similar clustering, like Legendre points (Chapter 17), have similarly good behavior. The explanation of this fact has a lot to do with potential theory, a subject we shall introduce in Chapter 12. Specifically, what makes Chebyshev or Legendre points effective is that each one has approximately the same average distance from the others, as measured in the sense of the geometric mean. On the interval $[-1, 1]$, this average distance is about $1/2$ (Exercise 2.6).

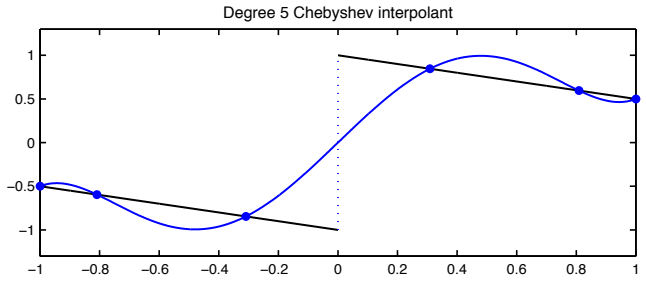
Chebfun is built on Chebyshev interpolants [Battles & Trefethen 2004]. For example, here is a certain step function:

```
x = chebfun('x'); f = sign(x) - x/2; plot(f,'k')
```



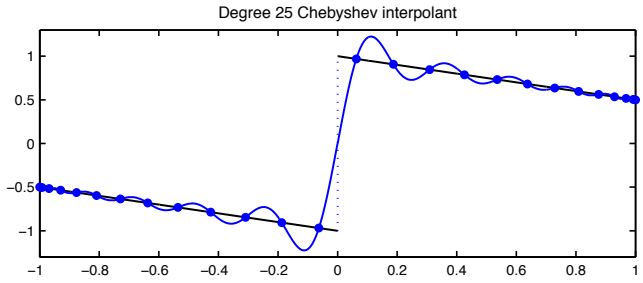
By calling `chebfun` with a second explicit argument of 6, we can construct the Chebyshev interpolant to f through 6 points, that is, of degree 5:

```
p = chebfun(f,6); hold on, plot(p,'.-')
```



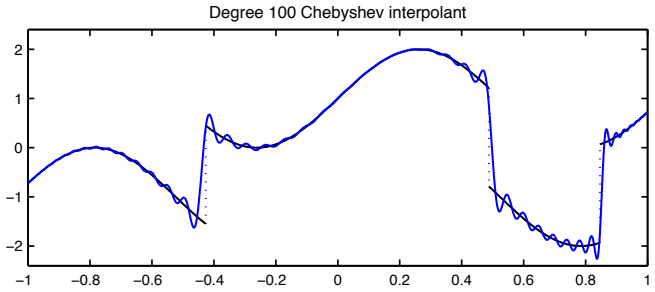
Similarly, here is the Chebyshev interpolant of degree 25:

```
plot(f,'k'), p = chebfun(f,26); hold on, plot(p,'.-')
```



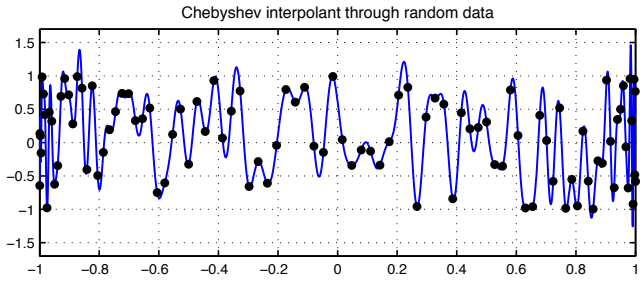
Here are a more complicated function and its interpolant of degree 100:

```
f = sin(6*x) + sign(sin(x+exp(2*x)));
plot(f,'k'), p = chebfun(f,101); hold on, plot(p)
```



Another way to use the `chebfun` command is by giving it an explicit vector of data rather than a function to sample, in which case it interprets the vector as data for a Chebyshev interpolant of the appropriate order. Here for example is the interpolant of degree 99 through random data values at 100 Chebyshev points in $[-1,1]$:

```
p = chebfun(2*rand(100,1)-1); plot(p,'-'), hold on, plot(p,'.k')
```



This experiment illustrates how robust Chebyshev interpolation is. If we had taken a million points instead of 100, the result would not have been much different mathematically, though it would have been harder to plot. We shall return to this figure in Chapter 15.

For illustrations like these it is interesting to pick data with jumps or wiggles, and Chapter 9 discusses such interpolants systematically. In applications where polynomial interpolants are most useful, however, the data will typically be smooth.

SUMMARY OF CHAPTER 2. *Polynomial interpolants in equispaced points in* $[-1,1]$ *have very poor approximation properties, but interpolants in Chebyshev points, which cluster near* ± 1*, are excellent.*

Exercise 2.1. Chebyshev interpolants through random data. (a) Repeat the experiment of interpolation through random data for 10, 100, 1000, and 10000 points. In each case use `minandmax(p)` to determine the minimum and maximum values of the interpolant and measure the computer time required for this computation (e.g. using `tic` and `toc`). You may find it helpful to increase Chebfun's standard plotting resolution with a command like `plot(p,'numpts',10000)`. (b) In addition to the four plots over $[-1,1]$, use `plot(p,'.-','interval',[0.9999 1])` to produce another plot of the 10000-point interpolant in the interval $[0.9999, 1]$. How many of the 10000 grid points fall in this interval?

Exercise 2.2. Limiting density as $n \to \infty$. (a) Suppose $x_0, \ldots, x_n$ are $n+1$ points equally spaced from -1 to 1. If $-1 \le a < b \le 1$, what fraction of the points fall in the interval $[a,b]$ in the limit $n \to \infty$? Give an exact formula. (b) Give the analogous formula for the case where $x_0, \ldots, x_n$ are the Chebyshev points. (c) How does the result of (b) match the number found in $[0.9999, 1]$ in the last exercise for the case $n = 9999$? (d) Show that in the limit $n \to \infty$, the density of the Chebyshev points near $x \in (-1,1)$ approaches $N/(\pi\sqrt{1-x^2})$ (see equation (12.10)).

Exercise 2.3. Rounding errors in computing Chebyshev points. On a computer in floating point arithmetic, the formula (2.2) for the Chebyshev points is not so good, because it lacks the expected symmetries. (a) Write a Matlab program that finds the smallest even value $n \ge 2$ for which, on your computer as computed by this formula, $x_{n/2} \ne 0$. (You will probably find that $n = 2$ is the first such value.) (b) Find the line in the code `chebpts.m` in which Chebfun computes Chebyshev points. What alternative formula does it use? Explain why this formula achieves perfect symmetry for all n in floating point arithmetic. (c) Show that this formula is mathematically equivalent to (2.2).

Exercise 2.4. Chebyshev points of the first kind. The Chebyshev points of the first kind, also known as *Gauss–Chebyshev points*, are obtained by taking the real parts of points on the unit circle mid-way between those we have considered, i.e. $x_j = \cos((j+\frac{1}{2})\pi/(n+1))$ for integers $0 \le j \le n$. Call `help chebpts` and `help legpts` to find out how to generate these points in Chebfun and how to generate Legendre points for comparison (these are roots of Legendre polynomials—see Chapter 17). For $n+1 = 100$, what is the maximum difference between a Chebyshev point of the first kind and the corresponding Legendre point? Draw a plot to illustrate as informatively as you can how close these two sets of points are.

Exercise 2.5. Convergence of Chebyshev interpolants. (a) Use Chebfun to produce a plot on a log scale of $\|f - p_n\|$ as a function of n for $f(x) = e^x$ on $[-1, 1]$, where p_n is the Chebyshev interpolant in $\mathcal{P}_n$. Take $\|\cdot\|$ to be the supremum norm, which can be computed by `norm(f-p,inf)`. How large must n be for accuracy at the level of machine precision? What happens if n is increased beyond this point? (b) The same questions for $f(x) = 1/(1 + 25x^2)$. Convergence rates like these will be analyzed in Chapters 7 and 8.

Exercise 2.6. Geometric mean distance between points. Write a code `meandistance` that takes as input a vector of points $x_0, \ldots, x_n$ in $[-1, 1]$ and produces a plot with x_j on the horizontal axis and the geometric mean of the distances of x_j to the other points on the vertical axis. (The Matlab command `prod` may be useful.) (a) What are the results for Chebyshev points with $n = 5, 10, 20$? (b) The same for Legendre points (see Exercise 2.4). (c) The same for equally spaced points from $x_0 = -1$ to $x_n = 1$.

Exercise 2.7. Chebyshev points scaled to the interval $[a, b]$. (a) Use `chebpts(10)` to print the values of the Chebyshev points in $[-1, 1]$ for $n = 9$. (b) Use `chebfun(@sin,10)` to compute the degree 9 interpolant $p(x)$ to $\sin(x)$ in these points. Make a plot showing $p(x)$ and $\sin(x)$ over the larger interval $[-6, 6]$, and also a semilog plot of $|f(x) - p(x)|$ over that interval. Comment on the results. (c) Now use `chebpts(10,[0 6])` to print the values of the Chebyshev points for $n = 9$ scaled to the interval $[0, 6]$. (d) Use `chebfun(@sin,[0 6],10)` to compute the degree 9 interpolant to $\sin(x)$ in these points, and make the same two plots as before over $[-6, 6]$. Comment.

3. Chebyshev polynomials and series

Throughout applied mathematics, one encounters three closely analogous canonical settings associated with the names of Fourier, Laurent, and Chebyshev. In fact, if we impose certain symmetries in the Fourier and Laurent cases, the analogies become equivalences. The Chebyshev setting is the one of central interest in this book, concerning a variable x and a function f defined on $[-1,1]$:

$$\textit{Chebyshev:} \quad x \in [-1,1], \quad f(x) \approx \sum_{k=0}^{n} a_k T_k(x). \tag{3.1}$$

Here T_k is the kth Chebyshev polynomial, which we shall discuss in a moment. For the equivalent Laurent problem, let z be a variable that ranges over the unit circle in the complex plane. Given $f(x)$, define a transplanted function $F(z)$ on the unit circle by the condition $F(z) = f(x)$, where $x = (z + z^{-1})/2$ as in (2.1). Note that this means that there are two values of z for each value of x, and F satisfies the symmetry property $F(z) = F(z^{-1})$. The series now involves a polynomial in both z and z^{-1}, known as a *Laurent polynomial:*

$$\textit{Laurent:} \quad |z| = 1, \quad F(z) = F(z^{-1}) \approx \frac{1}{2} \sum_{k=0}^{n} a_k (z^k + z^{-k}). \tag{3.2}$$

For the equivalent Fourier problem, let θ be a variable that ranges over $[-\pi, \pi]$, which we regard as a 2π-periodic domain. Transplant f and F to a function $\mathcal{F}$ defined on $[-\pi, \pi]$ by setting $\mathcal{F}(\theta) = F(e^{i\theta}) = f(\cos(\theta))$ as in (2.2). Now we have a 1-to-1 correspondence $z = e^{i\theta}$ between θ and z and a 2-to-1 correspondence between θ and x, with the symmetry $\mathcal{F}(\theta) = \mathcal{F}(-\theta)$, and the series is a

trigonometric polynomial:

$$\textit{Fourier:}\quad \theta \in [-\pi,\pi],\quad \mathcal{F}(\theta) = \mathcal{F}(-\theta) \approx \frac{1}{2}\sum_{k=0}^{n} a_k (e^{ik\theta} + e^{-ik\theta}). \tag{3.3}$$

One can carry (3.1)–(3.3) further by introducing canonical systems of grid points in the three settings. We have already seen the $(n+1)$-point Chebyshev grid,

$$\textit{Chebyshev points:}\quad x_j = \cos(j\pi/n),\quad 0 \le j \le n, \tag{3.4}$$

and we have interpreted these in terms of the $(2n)$th roots of unity:

$$\textit{Roots of unity:}\quad z_j = e^{ij\pi/n},\quad -n+1 \le j \le n. \tag{3.5}$$

These grids are transplants of the set of $2n$ equispaced points in $[-\pi,\pi]$:

$$\textit{Equispaced points:}\quad \theta_j = j\pi/n,\quad -n+1 \le j \le n. \tag{3.6}$$

All three of these settings are unassailably important. Real analysts cannot do without Fourier, complex analysts cannot do without Laurent, and numerical analysts cannot do without Chebyshev. Moreover, the mathematics of the connections between the three frameworks is beautiful. But all this symmetry presents an expository problem. Without a doubt, a fully logical treatment should consider x, z, and θ in parallel. Each theorem should appear in three forms. Each application should be one of a trio.

It was on this basis that I started to write a book in 2008. The symmetries were elegant, but as the chapters accumulated, I came to realize that this would be a very long book and not a lovable one. The excellent logic was just a dead weight. The next year, I started again with the decision that the book would focus on $x \in [-1,1]$. This is the setting closest to much of approximation theory and numerical analysis, and it has a further special feature: it is the one least familiar to people. Nobody is surprised if you compute a Fourier transform of a million points, but the fact that you can compute a polynomial interpolant through a million Chebyshev points surprises people indeed.

Here then is the mathematical plan for this book. Our central interest will be the approximation of functions $f(x)$ on $[-1,1]$. When it comes to deriving formulas and proving theorems, however, we shall generally transplant to $F(z)$ on the unit circle so as to make the tools of complex analysis most conveniently available.

Now let us turn to the definitions, already implicit in (3.1)–(3.3). The kth *Chebyshev polynomial* can be defined as the real part of the function z^k on the unit circle:

$$x = \tfrac{1}{2}(z + z^{-1}) = \cos\theta,\quad \theta = \cos^{-1} x, \tag{3.7}$$

$$T_k(x) = \tfrac{1}{2}(z^k + z^{-k}) = \cos(k\theta). \tag{3.8}$$

(Chebyshev polynomials were introduced by Chebyshev in the 1850s, though without the connection to the variables z and θ [Chebyshev 1854 & 1859]. The label T was apparently chosen by Bernstein, following French transliterations such as "Tchebischeff.") The Chebyshev polynomials are a family of orthogonal polynomials with respect to a certain weight function (Exercise 3.7), but we shall use orthogonality until Chapters 17–19.

It follows from (3.8) that T_k satisfies $-1 \le T_k(x) \le 1$ for $x \in [-1,1]$ and takes alternating values ± 1 at the $k+1$ Chebyshev points. What is not obvious is that T_k is a polynomial. We can verify this property by the computation

$$\tfrac{1}{2}(z+z^{-1})(z^k+z^{-k}) = \tfrac{1}{2}(z^{k+1}+z^{-k-1}) + \tfrac{1}{2}(z^{k-1}+z^{-k+1})$$

for any $k \ge 1$, that is,

$$2xT_k(x) = T_{k+1}(x) + T_{k-1}(x), \tag{3.9}$$

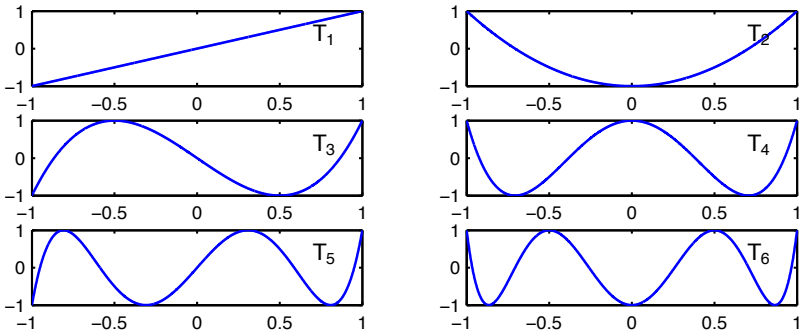
or in other words

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x). \tag{3.10}$$

By induction, this three-term recurrence relation implies that for each $k \ge 1$, T_k is a polynomial of degree exactly k with leading coefficient 2^{k-1}. In Chapters 18 and 19 the coefficients of this recurrence will be taken as the entries of a "colleague matrix," whose eigenvalues can be computed to find roots of polynomials or quadrature nodes.

The Chebfun command `chebpoly(n)` returns the chebfun corresponding to T_n.[1] Here for example are $T_1, \dots, T_6$:

```
for n = 1:6, T{n} = chebpoly(n); subplot(3,2,n), plot(T{n}), end
```



These plots do not show the Chebyshev points, which are the extremes of each curve: thus the numbers of Chebyshev points in the six plots are 2, 3, 4, 5, 6, and 7.

Here are the coefficients of these polynomials with respect to the monomial basis $1, x, x^2, \dots$. As usual, Matlab orders coefficients from highest degree down to degree zero.

[1] The name of the software system is Chebfun, with a capital C. A representation of a particular function in Chebfun is called a chebfun, with a lower-case c.

```
for n = 1:6, disp(poly(T{n})), end
```

```
     1     0
     2     0    -1
     4     0    -3     0
     8     0    -8     0     1
    16     0   -20     0     5     0
    32     0   -48     0    18     0    -1
```

So, for example,

$$T_5(x) = 16x^5 - 20x^3 + 5x.$$

The monomial basis is familiar and comfortable, but you should never use it for numerical work with functions on an interval. Use the Chebyshev basis instead (Exercise 3.8). (If the domain is $[a, b]$ rather than $[-1, 1]$, the Chebyshev polynomials must be scaled accordingly, and Chebfun does this automatically when it works on other intervals.) For example, x^5 has the Chebyshev expansion

$$x^5 = \frac{5}{80}T_5(x) + \frac{5}{16}T_3(x) + \frac{5}{8}T_1(x).$$

We can calculate such expansion coefficients by using the command `chebpoly(p)`, where p is the chebfun whose coefficients we want to know:

```
x = chebfun('x'); chebpoly(x.^5)

ans =
    0.0625         0    0.3125         0    0.6250         0
```

Any polynomial p can be written uniquely like this as a finite Chebyshev series: the functions $T_0(x), T_1(x), \ldots, T_n(x)$ form a basis for $\mathcal{P}_n$. Since p is determined by its values at Chebyshev points, it follows that there is a one-to-one linear mapping between values at Chebyshev points and Chebyshev expansion coefficients. This mapping can be applied in $O(n \log n)$ operations with the aid of the Fast Fourier Transform (FFT) or the Fast Cosine Transform, a crucial observation for practical work that was perhaps first made by Ahmed and Fisher and Orzsag around 1970 [Ahmed & Fisher 1970, Orszag 1971a and 1971b, Gentleman 1972b, Geddes 1978]. This is what Chebfun does every time it constructs a chebfun. We shall not give details of the FFT.

Just as a polynomial p has a finite Chebyshev series, a more general function f has an infinite Chebyshev series. Exactly what kind of "more general function" can we allow? For an example like $f(x) = e^x$ with a rapidly converging Taylor series, everything will surely be straightforward, but what if f is merely differentiable rather than analytic? Or what if it is continuous but not differentiable? Analysts have studied such cases carefully, identifying exactly what degrees of smoothness correspond to what kinds of convergence of Chebyshev series. We shall not concern ourselves with trying to state the sharpest possible result but will just make a particular assumption that covers most applications.

We shall assume that f is *Lipschitz continuous* on $[-1,1]$. Recall that this means that there is a constant C such that $|f(x) - f(y)| \le C|x-y|$ for all $x, y \in [-1,1]$. Recall also that a series is *absolutely convergent* if it remains convergent when each term is replaced by its absolute value, and that this implies that one can reorder the terms arbitrarily without changing the result. Such matters are discussed in analysis textbooks such as [Rudin 1976].

Here is our basic theorem about Chebyshev series and their coefficients.

I think this is also referred to as Chebyshev expansion of a polynomial

Theorem 3.1. Chebyshev series. *If f is Lipschitz continuous on $[-1,1]$, it has a unique representation as a Chebyshev series,*

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x), \tag{3.11}$$

which is absolutely and uniformly convergent, and the coefficients are given for $k \ge 1$ by the formula

$$a_k = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_k(x)}{\sqrt{1-x^2}} dx, \tag{3.12}$$

and for $k = 0$ by the same formula with the factor $2/\pi$ changed to $1/\pi$.

Proof. Equation (3.12) will come from the Cauchy integral formula, and to make this happen, we begin by transplanting f to F on the unit circle as described above: $F(z) = F(z^{-1}) = f(x)$ with $x = (z + z^{-1})/2$. To convert between integrals in x and z, we have to convert between dx and dz:

$$dx = \tfrac{1}{2}(1 - z^{-2})\, dz = \tfrac{1}{2} z^{-1}(z - z^{-1})\, dz.$$

Since

$$\tfrac{1}{2}(z - z^{-1}) = i \operatorname{Im} z = \pm i \sqrt{1-x^2},$$

this implies

$$dx = \pm i z^{-1} \sqrt{1-x^2}\, dz.$$

In these equations the plus sign applies for $\operatorname{Im} z \ge 0$ and the minus sign for $\operatorname{Im} z \le 0$.

These formulas have implications for smoothness. Since $\sqrt{1-x^2} \le 1$ for all $x \in [-1,1]$, they imply that if $f(x)$ is Lipschitz continuous, then so is $F(z)$. By a standard result in Fourier analysis, this implies that F has a unique representation as an absolutely and uniformly convergent Laurent series on the unit circle,

$$F(z) = \frac{1}{2} \sum_{k=0}^{\infty} a_k (z^k + z^{-k}) = \sum_{k=0}^{\infty} a_k T_k(x).$$

Recall that a *Laurent series* is an infinite series in both positive and negative powers of z, and that if F is analytic, such a series converges in the interior of an annulus. A good treatment of Laurent series for analytic functions can be found in [Markushevich 1985]; see also other complex variables texts such as [Hille 1973, Priestley 2003, Saff & Snider 2003].

The kth Laurent coefficient of a Lipschitz continuous function $G(z) = \sum_{k=-\infty}^{\infty} b_k z^k$ on the unit circle can be computed by the Cauchy integral formula,

$$b_k = \frac{1}{2\pi i} \int_{|z|=1} z^{-1-k} G(z)\, dz.$$

(We shall make more substantial use of the Cauchy integral formula in Chapters 8 and 11–12.) The notation $|z| = 1$ indicates that the contour consists of the unit circle traversed once in the positive (counterclockwise) direction. Here we have a function F with the special symmetry property $F(z) = F(z^{-1})$, and we have also introduced a factor $1/2$ in front of the series. Accordingly, we can compute the coefficients a_k from either of two contour integrals,

$$a_k = \frac{1}{\pi i} \int_{|z|=1} z^{-1+k} F(z)\, dz = \frac{1}{\pi i} \int_{|z|=1} z^{-1-k} F(z)\, dz, \tag{3.13}$$

with πi replaced by $2\pi i$ for $k = 0$.

In particular, we can get a formula for a_k that is symmetric in k and $-k$ by combining the two integrals like this:

$$a_k = \frac{1}{2\pi i} \int_{|z|=1} (z^{-1+k} + z^{-1-k}) F(z)\, dz = \frac{1}{\pi i} \int_{|z|=1} z^{-1}\, T_k(x) F(z)\, dz, \tag{3.14}$$

with πi replaced by $2\pi i$ for $k = 0$. Replacing $F(z)$ by $f(x)$ and $z^{-1}dz$ by $-i\, dx/(\pm\sqrt{1-x^2})$ gives

$$a_k = -\frac{1}{\pi} \int_{|z|=1} \frac{f(x) T_k(x)}{\pm\sqrt{1-x^2}}\, dx,$$

with π replaced by 2π for $k = 0$. We have now almost entirely converted to the x variable, except that the contour of integration is still the circle $|z| = 1$. When z traverses the circle all the way around in the positive direction, x decreases from 1 to -1 and then increases back to 1 again. At the turning point $z = x = -1$, the $\pm$ sign attached to the square root switches from $+$ to $-$. Thus instead of cancelling, the two traverses of $x \in [-1, 1]$ contribute equal halves to a_k. Converting to a single integration from -1 to 1 in the x variable multiplies the integral by $-1/2$, hence multiplies the formula for a_k by -2, giving (3.12). ∎

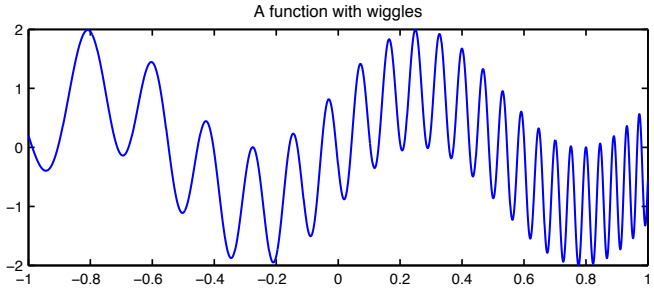
We now know that any function f, so long as it is Lipschitz continous, has a Chebyshev series. Chebfun represents a function as a finite series of some degree n, storing both its values at Chebyshev points and also, equivalently, their Chebyshev coefficients. How does it figure out the right value of n? Given a set of $n + 1$ samples, it converts the data to a Chebyshev expansion of degree n and examines the resulting Chebyshev coefficients. If several of these in a row fall below a relative level of approximately 10^{-15}, then the grid is judged to be fine enough. For example, here are the Chebyshev coefficients of the chebfun corresponding to e^x:

```
f = exp(x); a = chebpoly(f); a(end:-1:1)'

ans =
   1.266065877752008
   1.130318207984970
   0.271495339534077
   0.044336849848664
   0.005474240442094
   0.000542926311914
   0.000044977322954
   0.000003198436462
   0.000000199212481
   0.000000011036772
   0.000000000550590
   0.000000000024980
   0.000000000001039
   0.000000000000040
   0.000000000000001
```

Notice that the last coefficient is about at the level of machine precision.

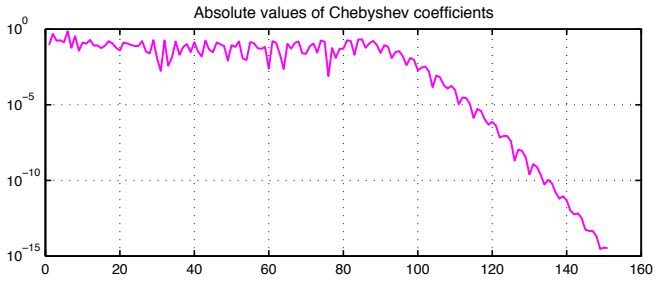
For complicated functions it is often more interesting to plot the coefficients than to list them. For example, here is a function with a number of wiggles:

```
f = sin(6*x) + sin(60*exp(x)); plot(f)
```



If we plot the absolute values of the Chebyshev coefficients, here is what we find:

```
a = chebpoly(f); semilogy(abs(a(end:-1:1)),'m')
```



One can explain this plot as follows. Up to degree about $k = 80$, a Chebyshev series cannot resolve f much at all, for the oscillations occur on too short wavelengths. After that, the series begins to converge rapidly. By the time we reach $k = 150$, the accuracy is about 15 digits, and the computed Chebyshev series is truncated there. We can find out exactly where the truncation took place with the command `length(f)`:

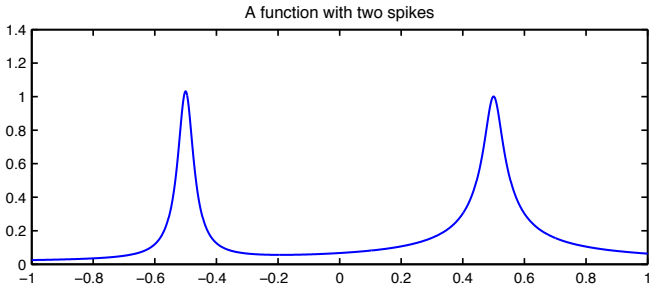
```
length(f)

ans = 151
```

This tells us that the chebfun is a polynomial interpolant through 151 points, that is, of degree 150.

Without giving all the engineering details, here is a fuller description of how Chebfun constructs its approximation. First it calculates the polynomial interpolant through the function sampled at 9 Chebyshev points, i.e., a polynomial of degree 8, and checks whether the Chebyshev coefficients appear to be small enough. For the example just given, the answer is no. Then it tries 17 points, then 33, then 65, and so on. In this case Chebfun judges at 257 points that the Chebyshev coefficients have fallen to the level of rounding error. At this point it truncates the tail of terms deemed to be negligible, leaving a series of 151 terms (Exercise 3.13). The corresponding degree 150 polynomial is then evaluated at 151 Chebyshev points via FFT, and these 151 numbers become the data defining this particular chebfun. Engineers would say that the signal has been *downsampled* from 257 points to 151.

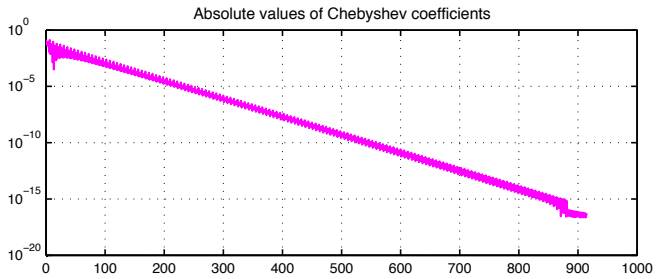
For another example we consider a function with two spikes:

```
f = 1./(1+1000*(x+.5).^2) + 1./sqrt(1+1000*(x-.5).^2); plot(f)
```



Here are the Chebyshev coefficients of the chebfun. This time, instead of `chebpoly` and `semilogy`, we execute the special command `chebpolyplot`, which has the same effect.

```
chebpolyplot(f,'m')
```



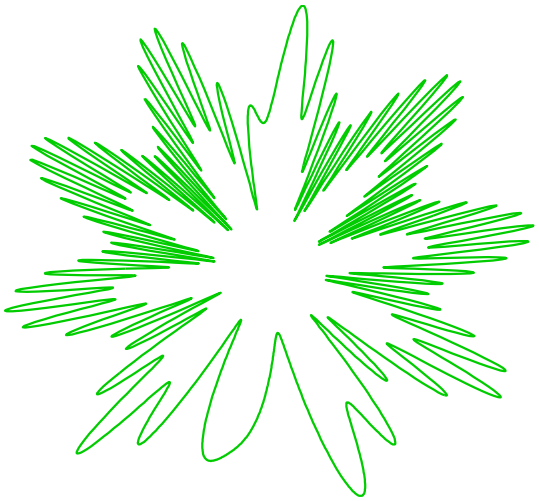
Note that although it is far less wiggly, this function needs six times as many points to resolve as the previous one (Exercise 3.13). We shall explain these polynomial degrees in Chapter 8.

Chebyshev interpolants are effective for complex functions (still defined on a real interval) as well as real ones. Here, for example, is a complex function that happens to be periodic, though the Chebyshev representation does not take advantage of this fact.

```
f = (3+sin(10*pi*x)+sin(61*exp(.8*sin(pi*x)+.7))).*exp(1i*pi*x);
```

A plot shows the image of $[-1,1]$ under f, which appears complicated:

```
plot(f,'color',[0 .8 0])
```



Yet the degree of the polynomial is not so high:

```
length(f)
```

```
ans = 617
```

People often ask, is there anything special about Chebyshev points and Chebyshev polynomials? Could we equally well interpolate in other points and expand in other sets of polynomials? From an approximation point of view, the answer is yes, and in particular, Legendre points and Legendre polynomials have much the same power for representing a general function f, as we shall see in Chapters 17–19. Legendre points and polynomials are neither much better than Chebyshev for approximating functions, nor much worse; they are essentially the same. One can improve upon both Legendre and Chebyshev, shrinking the number of sample points needed to represent a given function by a factor of up to $\pi/2$, but to do so one must leave the class of polynomials. See Chapter 22.

Nevertheless, there is a big advantage of Chebyshev over Legendre points, and this is that one can use the FFT to go from point values to coefficients and back again. There are algorithms that make such computations practicable for Legendre interpolants too [Piessens 1974, Alpert & Rokhlin 1991, Dutt, Gu & Rokhlin 1996, Potts, Steidl & Tasche 1998, Iserles 2011]—see also Theorem 19.6 of this book—but Chebyshev remains the easiest case.

Summary of Chapter 3. *The Chebyshev polynomial $T_k(x)$ is an analogue for $[-1,1]$ of the monomial z^k on the unit circle. Each Lipschitz continuous function f on $[-1,1]$ has an absolutely and uniformly convergent Chebyshev series, that is, an expansion $f(x) = a_0T_0(x) + a_1T_1(x) + \ldots$.*

Exercise 3.1. Monomial and Chebyshev coefficients. Let $p \in \mathcal{P}_n$ have coefficient vectors $a = (a_0, a_1, \ldots, a_n)^T$ for a Chebyshev series and $b = (b_0, b_1, \ldots, b_n)^T$ for a series in the monomials $1, x, \ldots, x^n$. Show that a and b are related by $Aa = b$, where A is an upper-triangular matrix, whose entries you should describe precisely, though you don't have to give explicit formulas for them. Prove that any $p \in \mathcal{P}_n$ has uniquely defined coefficient vectors a and b for both representations.

Exercise 3.2. A Chebyshev coefficient. Use Chebfun to determine numerically the coefficient of T_5 in the Chebyshev expansion of $\tan^{-1}(x)$ on $[-1,1]$.

Exercise 3.3. Chebyshev coefficients and "rat". (a) Use Chebfun to determine numerically the coefficients of the Chebyshev series for $1 + x^3 + x^4$. By inspection, identify these rational numbers. Use the Matlab command `[n,d] = rat(c)` to confirm this. (b) Use Chebfun and `rat` to make good guesses as to the Chebyshev coefficients of $x^7/7 + x^9/9$. (Of course it is not hard to figure them out analytically.)

Exercise 3.4. Dependence on wave number. (a) Calculate the length $L(k)$ of the chebfun corresponding to $f(x) = \sin(kx)$ on $[-1,1]$ for $k = 1, 2, 4, 8, \ldots, 2^{10}$. (You can do this elegantly by defining a Matlab anonymous function `f = @(k)`....) Make a loglog plot of $L(k)$ as a function of k and comment on the result. (b) Do the same for $g(x) = 1/(1 + (kx)^2)$.

Exercise 3.5. Chebyshev series of a complicated function. (a) Make chebfuns of the three functions $f(x) = \tanh(x)$, $g(x) = 10^{-5}\tanh(10x)$, $h(x) = 10^{-10}\tanh(100x)$ on $[-1,1]$, and call `chebpolyplot` to show their Chebyshev coefficients. Comment on the results. (b) Now define $s = f + g + h$ and comment on the result of `chebpolyplot` applied to s. Chebfun does not automatically chop the tail of

a Chebyshev series obtained by summation, but applying the `simplify` command will do this. What happens with `chebpolyplot(simplify(s))`?

Exercise 3.6. Chebyshev series of sign(x) and $|x|$ [Bernstein 1914]. Derive the following Chebyshev series coefficients by using the first equality in (3.14). (a) For $f(x) = \text{sign}(x)$, $a_k = 0$ for k even and $a_k = (4/\pi)(-1)^{k-1}/k$ for k odd. (b) For $f(x) = |x|$, $a_k = 0$ for k odd, $a_0 = 2/\pi$, and $a_k = (4/\pi)(-1)^{(k/2)}/(1-k^2)$ for $k \geq 2$ even.

Exercise 3.7. Orthogonality of Chebyshev polynomials. Equation (3.12) gives the Chebyshev coefficient a_k of f by integration of f against just the single Chebyshev polynomial T_k. This formula implies an orthogonality property for $\{T_j\}$ involving a weighted integral. State exactly what this orthogonality property is and show carefully how it follows from the equations of this chapter.

Exercise 3.8. Conditioning of the Chebyshev basis. Although the Chebyshev polynomials are not orthogonal with respect to the standard unweighted inner product, they are close enough to orthogonal to provide a well-behaved basis. Set `T = chebpoly(0:10)` and explore the Chebfun "quasimatrix" that results with commands like `size(T)`, `spy(T)`, `plot(T)`, `svd(T)`. Explain the meaning of `T` (you may find Chapter 6 of the Chebfun Guide helpful) and determine the condition number of this basis with `cond(T)`. (b) Now construct the corresponding quasimatrix of monomials by executing `x = chebfun('x'); M = T; for j = 0:10, M(:,j+1) = x.^j; end`. What is the condition number of `M`? (c) Produce a plot of these two condition numbers for quasimatrices whose columns span $\mathcal{P}_n$ over $[-1,1]$ for $n = 0, 1, \ldots, 10$. (d) What happens to the condition numbers if M is constructed from monomials on $[0,1]$ rather than $[-1,1]$ via `x = chebfun('x',[0,1])`?

Exercise 3.9. Derivatives at endpoints. Prove from (3.10) that the derivatives of the Chebyshev polynomials satisfy $T_n'(1) = n^2$ for each $n \geq 0$. (*Markov's inequality* asserts that for any $p \in \mathcal{P}_n$, $\|p'\| \leq n^2\|p\|$, where $\|\cdot\|$ is the supremum norm.)

Exercise 3.10. Odd and even functions. Show that if f is an odd function on $[-1,1]$, its Chebyshev coefficients of even order are zero; show similarly that if f is even, its odd order coefficients are zero.

Exercise 3.11. A function neither even nor odd. Apply `chebpolyplot` to the chebfun for $f(x) = \exp(x)/(1+10000x^2)$. Why does the plot have the appearance of a stripe?

Exercise 3.12. Extrema and roots of Chebyshev polynomials. Give formulas for the extrema and roots of T_n in $[-1,1]$.

Exercise 3.13. Chebyshev coefficients and machine precision. By a command like `f = chebfun('exp(x)',np)`, one can force Chebfun to produce a chebfun of length `np` (i.e., degree `np`-1) rather than determine the length automatically. (a) Do this for the "function with wiggles" of this section with `np` $= 257$, and comment on how the `chebpolyplot` result differs from that shown in the text. (b) Likewise for the "function with two spikes" with `np` $= 2049$.

Exercise 3.14. Chebyshev series for a simple pole. (a) Let t be a complex number with $|t| < 1$ and define $F(z) = (z-t)^{-1} + (z^{-1}-t)^{-1}$. What is the Laurent series for F? (b) For the same t, show further that

$$1 + 2\sum_{k=1}^{\infty} t^k T_k(x) = \frac{1-t^2}{1-2tx+t^2}. \tag{3.15}$$

(This formula can be interpreted as a generating function for the Chebyshev polyno-

mials.) (c) Let $a \notin [-1,1]$ be a real or complex number and let t be a real or complex number with $|t| < 1$ such that $(t + t^{-1})/2 = a$. Show that

$$\frac{1}{x-a} = \frac{2}{t-t^{-1}} \left[1 + 2 \sum_{k=1}^{\infty} t^k T_k(x) \right]. \tag{3.16}$$

Exercise 3.15. Chebyshev series of e^{ax}. It can be shown that the Chebyshev series of e^{ax} is

$$e^{ax} = 2 \sum_{k=0}^{\infty}{}' I_k(a) T_k(x), \tag{3.17}$$

where I_k is the modified Bessel function of the first kind and the prime indicates that the term $k = 0$ is to be multiplied by $1/2$. Derive the Chebyshev series for $\sinh(ax)$ and $\cosh(ax)$.

Exercise 3.16. Clenshaw's algorithm. Let a polynomial $p \in \mathcal{P}_n$ be given by a finite Chebyshev series (3.11) and let $x \in [-1,1]$ be given. Show that $p(x)$ can be evaluated by the following process. Set $u_{n+1} = 0$ and $u_n = a_n$ and

$$u_k = 2x\,u_{k+1} - u_{k+2} + a_k, \quad k = n-1, n-2, \ldots, 0. \tag{3.18}$$

Then $p(x) = \frac{1}{2}(a_0 + u_0 - u_2)$.

4. Interpolants, projections, and aliasing

Suppose $f(x)$ is a Lipschitz continuous function on $[-1,1]$ with Chebyshev series coefficients $\{a_k\}$ as in Theorem 3.1,

$$f(x) = \sum_{k=0}^{\infty} a_k T_k(x). \tag{4.1}$$

One approximation to f in $\mathcal{P}_n$ is the polynomial obtained by *interpolation* in Chebyshev points:

$$p_n(x) = \sum_{k=0}^{n} c_k T_k(x). \tag{4.2}$$

Another is the polynomial obtained by *truncation* or *projection* of the series to degree n, whose coefficients through degree n are the same as those of f itself:

$$f_n(x) = \sum_{k=0}^{n} a_k T_k(x). \tag{4.3}$$

The relationship of the Chebyshev coefficients of f_n to those of f is obvious, and in a moment we shall see that the Chebyshev coefficients of p_n have simple expressions too. In computational work generally, and in particular in Chebfun, the polynomials $\{p_n\}$ are usually almost as good approximations to f as the polynomials $\{f_n\}$, and easier to work with, since one does not need to evaluate the integral (3.12). The polynomials $\{f_n\}$, on the other hand, are also interesting. In this book, most of our computations will make use of $\{p_n\}$, but many of our theorems will treat both cases. A typical example is Theorem 8.2, which

asserts that if f is analytic on $[-1,1]$, then both $\|f-f_n\|$ and $\|f-p_n\|$ decrease geometrically to 0 as $n\to\infty$.

The key to understanding $\{c_k\}$ is the phenomenon of *aliasing*, a term that originated with radio engineers early in the 20th century. On the $(n+1)$-point Chebyshev grid, it is obvious that any function f is indistinguishable from a polynomial of degree n. But something more is true: any Chebyshev polynomial T_N, no matter how big N is, is indistinguishable on the grid from a *single* Chebyshev polynomial T_m for some m with $0\le m\le n$. We state this as a theorem.

Theorem 4.1. Aliasing of Chebyshev polynomials. *For any $n\ge 1$ and $0\le m\le n$, the following Chebyshev polynomials take the same values on the $(n+1)$-point Chebyshev grid:*

$$T_m,\ T_{2n-m},\ T_{2n+m},\ T_{4n-m},\ T_{4n+m},\ T_{6n-m},\dots.$$

Equivalently, for any $k\ge 0$, T_k takes the same value on the grid as T_m with

$$m = |(k+n-1)(\mathrm{mod}\, 2n) - (n-1)|, \tag{4.4}$$

a number in the range $0\le m\le n$.

Proof. Recall from (2.1) and (3.8) that Chebyshev polynomials on $[-1,1]$ are related to monomials on the unit circle by $T_m(x) = (z^m + z^{-m})/2$, and Chebyshev points are related to $(2n)$th roots of unity by $x_m = (z_m + z_m^{-1})/2$. It follows that the first assertion of the theorem is equivalent to the statement that the following functions take the same values at the $(2n)$th roots of unity:

$$z^m + z^{-m},\ z^{2n-m} + z^{m-2n},\ z^{2n+m} + z^{-2n-m},\dots.$$

Inspection of the exponents shows that in every case, modulo $2n$, we have one exponent equal to $+m$ and the other to $-m$. The conclusion now follows from the elementary phenomenon of aliasing of monomials on the unit circle: at the $(2n)$th roots of unity, $z^{2\nu n}=1$ for any integer ν.

For the second assertion (4.4), suppose first that $0\le k\,(\mathrm{mod}\, 2n)\le n$. Then $n-1\le (k+n-1)(\mathrm{mod}\, 2n)\le 2n-1$, so (4.4) reduces to $m = k\,(\mathrm{mod}\, 2n)$, with $0\le m\le n$, and we have just shown that this implies that T_k and T_m take the same values on the grid. On the other hand, suppose that $n+1\le k\,(\mathrm{mod}\, 2n)\le 2n-1$. Then $0\le (k+n-1)(\mathrm{mod}\, 2n)\le n-2$, so the absolute value becomes a negation and (4.4) reduces to $m = -k\,(\mathrm{mod}\, 2n)$, with $1\le m\le n$. Again we have just shown that this implies that T_k and T_m take the same values on the grid. ∎

Here is a numerical illustration of Theorem 4.1. Taking $n=4$, let `X` be the Chebyshev grid with $n+1$ points, and let $T\{1\},\dots,T\{10\}$ be the first ten Chebyshev polynomials:

```
n = 4; X = chebpts(n+1); for k = 1:10, T{k} = chebpoly(k); end
```

Then T_3 and T_5 are the same on the grid:

```
disp([T{3}(X) T{5}(X)])
```

```
T3                  T5
   -1.000000000000000  -1.000000000000000
    0.707106781186548   0.707106781186547
                    0                   0
   -0.707106781186548  -0.707106781186547
    1.000000000000000   1.000000000000000
```

So are T_1, T_7, and T_9:

```
disp([T{1}(X) T{7}(X) T{9}(X)])
```

```
T1                  T7                  T9
   -1.000000000000000  -1.000000000000000  -1.000000000000000
   -0.707106781186547  -0.707106781186548  -0.707106781186547
                    0                   0                   0
    0.707106781186547   0.707106781186548   0.707106781186547
    1.000000000000000   1.000000000000000   1.000000000000000
```

As a corollary of Theorem 4.1, we can now derive the connection between $\{a_k\}$ and $\{c_k\}$. The following result can be found in [Clenshaw & Curtis 1960].

Theorem 4.2. Aliasing formula for Chebyshev coefficients. *Let f be Lipschitz continuous on $[-1,1]$, and let p_n be its Chebyshev interpolant in $\mathcal{P}_n$, $n \geq 1$. Let $\{a_k\}$ and $\{c_k\}$ be the Chebyshev coefficients of f and p_n, respectively. Then*

$$c_0 = a_0 + a_{2n} + a_{4n} + \cdots, \tag{4.5}$$

$$c_n = a_n + a_{3n} + a_{5n} + \cdots, \tag{4.6}$$

and for $1 \leq k \leq n-1$,

$$c_k = a_k + (a_{k+2n} + a_{k+4n} + \cdots) + (a_{-k+2n} + a_{-k+4n} + \cdots). \tag{4.7}$$

Proof. By Theorem 3.1, f has a unique Chebyshev series (3.11), and it converges absolutely. Thus we can rearrange the terms of the series without affecting convergence, and in particular, each of the three series expansions written above converges since they correspond to the Chebyshev series (3.11) evaluated at $x = 1$. So the formulas (4.5)–(4.7) do indeed define certain numbers $c_0, \ldots, c_n$. Taking these numbers as coefficients multiplied by the corresponding Chebyshev polynomials $T_0, \ldots, T_n$ gives us a polynomial of degree n. By Theorem 4.1, this polynomial takes the same values as f at each point of the Chebyshev grid. Thus it is the unique interpolant $p_n \in \mathcal{P}_n$. ∎

We can summarize Theorem 4.2 as follows. On the $(n+1)$-point grid, any function f is indistinguishable from a polynomial of degree n. In particular, the Chebyshev series of the polynomial interpolant to f is obtained by reassigning

all the Chebyshev coefficients in the infinite series for f to their aliases of degrees 0 through n.

As a corollary, Theorems 4.1 and 4.2 give us absolutely convergent series for $f - f_n$ and $f - p_n$, which we shall exploit in Chapters 7 and 8:

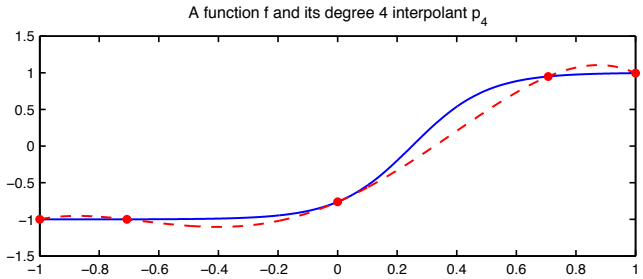
$$f(x) - f_n(x) = \sum_{k=n+1}^{\infty} a_k T_k(x), \tag{4.8}$$

$$f(x) - p_n(x) = \sum_{k=n+1}^{\infty} a_k (T_k(x) - T_m(x)), \tag{4.9}$$

where $m = m(k, n)$ is given by (4.4).

To illustrate Theorem 4.2, here is the function $f(x) = \tanh(4x - 1)$ (solid) and its degree 4 Chebyshev interpolant $p_4(x)$ (dashed):

```
x = chebfun('x'); f = tanh(4*x-1); n = 4;
pn = chebfun(f,n+1); plot(f), hold on, plot(pn,'.--r')
```



The first 5 Chebyshev coefficients of f,

```
a = chebpoly(f); a = a(end:-1:1)'; a(1:n+1)

ans =
  -0.166584582703135
   1.193005991160944
   0.278438064117869
  -0.239362401056012
  -0.176961398392888
```

are different from the Chebyshev coefficients of p_n,

```
c = chebpoly(pn); c = c(end:-1:1)'

c =
  -0.203351068209675
   1.187719968517890
   0.379583465333916
  -0.190237989543227
  -0.178659622412173
```

As asserted in (4.5) and (4.6), the coefficients c_0 and c_n are given by sums of coefficients a_k with a stride of $2n$:

```
c0 = sum(a(1:2*n:end)), cn = sum(a(n+1:2*n:end))
```

```
c0 = -0.203351068209675
cn = -0.178659622412174
```

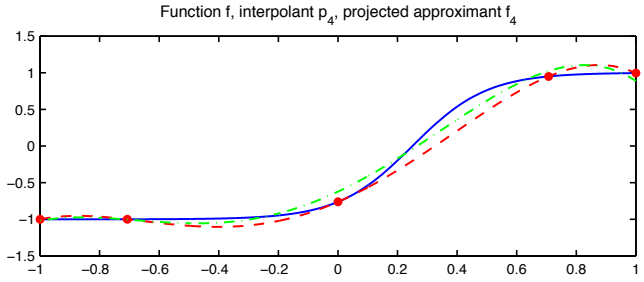
And as asserted in (4.7), the coefficients c_1 through c_{n-1} involve two sums of this kind:

```
for k = 1:n-1
  ck = sum(a(1+k:2*n:end)) + sum(a(1-k+2*n:2*n:end))
end
```

```
ck = 1.187719968517889
ck = 0.379583465333916
ck = -0.190237989543227
```

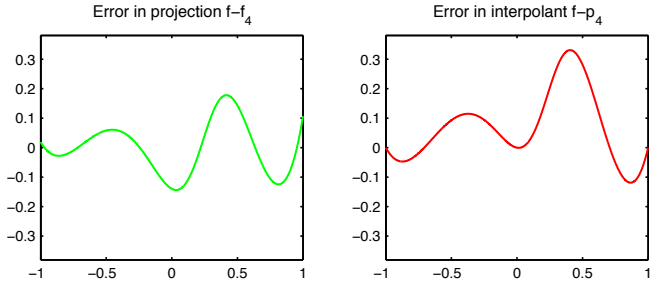
Following up on the last figure, how does the truncated series f_n compare with the interpolant p_n as an approximation to f? Chebfun includes a `'trunc'` option for computing f_n, which we now add to the plot as a dot-dash line:

```
fn = chebfun(f,'trunc',n+1); plot(fn,'-.g')
```



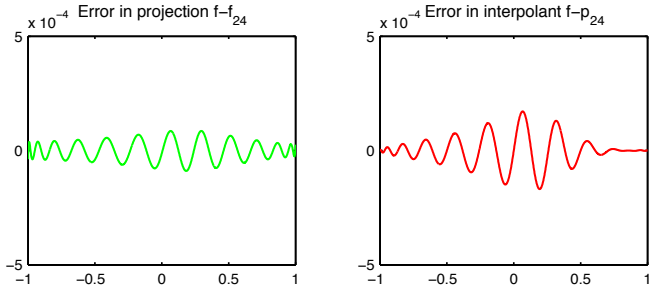
Here are the errors $f - f_n$ and $f - p_n$:

```
subplot(1,2,1), plot(f-fn,'g'), subplot(1,2,2), plot(f-pn,'r')
```



Here is the analogous plot with $n = 4$ increased to 24:

```
n = 24; pn = chebfun(f,n+1); fn = chebfun(f,'trunc',n+1);
subplot(1,2,1), plot(f-fn,'g'), subplot(1,2,2), plot(f-pn,'r')
```



On the basis of plots like these, one might speculate that f_n may often be a better approximation than p_n, but that the difference is small. This is indeed the case, as we shall confirm in Theorems 7.2 and 8.2, both of which suggest a difference of a factor of 2, and Theorem 16.1, which suggests a factor of $\pi/2$.

Let us review where we stand. We have considered Chebyshev interpolants (Chapter 2) and Chebyshev expansions (Chapter 3) for a Lipschitz continuous function $f(x)$ defined on $[-1, 1]$. Mathematically speaking, each coefficient of a Chebyshev expansion is equal to the value of the integral (3.12). This formula, however, is not needed for effective polynomial approximation, since Chebyshev interpolants are nearly as accurate as projections. Chebfun readily computes Chebyshev coefficients of polynomial interpolants, and this is done not by evaluating the integral but by taking the FFT of the sample values in Chebyshev points. If the degree of the interpolant is high enough that the polynomial matches f to machine precision, then the Chebyshev coefficients will usually match too.

Summary of Chapter 4. *Two excellent methods of approximating a function f on $[-1, 1]$ by a polynomial are truncation of its Chebyshev series, also known as projection, and interpolation in Chebyshev points. The Chebyshev interpolant is the polynomial obtained by reassigning contributions of degree $> n$ in the Chebyshev series to their aliases of degree $\le n$. The two approximations are typically within a factor of 2 of each other in accuracy.*

Exercise 4.1. Node polynomial for Chebyshev points. Show using Theorem 4.1 that $p(x) = 2^{-n}(T_{n+1}(x) - T_{n-1}(x))$ is the unique monic polynomial in $\mathcal{P}_{n+1}$ with zeros at the $n + 1$ Chebyshev points (2.2).

Exercise 4.2. Examples of aliasing. (a) On the $(n + 1)$-point Chebyshev grid with $n = 20$, which Chebyshev polynomials T_k take the same values as T_5? (b) Use Chebfun to draw plots illustrating some of these intersections.

Exercise 4.3. Aliasing in roots of unity. For each $n \geq 0$, let $p_n \in \mathcal{P}_n$ be the degree n polynomial interpolant to the function $f(z) = z^{-1}$ at the $(n+1)$st roots of unity on the unit circle in the z-plane. Use the aliasing observation of the proof of Theorem 4.1 to prove that in the closed unit disk of complex numbers z with $|z| \leq 1$, there is one and only one value z for which p_n converges to f as $n \to \infty$. (This example comes from [Méray 1884].)

Exercise 4.4. Fooling the Chebfun constructor. (a) Construct the Matlab anonymous function `f = @(M) chebfun(@(x) 1+exp(-(M*(x-0.4)).^4))` and plot `f(10)` and `f(100)`. This function has a narrow spike of width proportional to $1/M$. Confirm this by comparing `sum(f(10))` and `sum(f(100))`. (b) Plot `length(f(M))` as a function of M for $M = 1, 2, 3, \ldots$, going into the region where the length becomes 1. What do you think is happening? (c) Let `Mmax` be the largest integer for which the constructor behaves normally and execute `semilogy(f(Mmax)-1,'interval',[.3 .5])`. Superimpose on this plot information to show the locations of the points returned by `chebpts(9)`, which is the default initial grid on which Chebfun samples a function. Explain how this result fits with (b). (d) Now for `np` taking values 17, 33, 65, 129, execute `chebfunpref('minsamples',np)` and `length(f(np))`, and plot the Chebyshev points on your semilog plot of (c). The `minsamples` flag forces Chebfun to sample the function at the indicated number of points. How do these results match your observations of (b) and (c)? When you're done, be sure to return Chebfun to its default state with `chebfunpref('factory')`.

Exercise 4.5. Relative precision. Try Exercise 4.4 again but without the "1+" in the definition of `f`. The value of `Mmax` will be different, and the reason has to do with Chebfun's aim of constructing each function to about 15 digits of relative precision, not absolute. Can you figure out what is happening and explain it quantitatively?

Exercise 4.6. Chebfun computation of truncations. In the text we computed Chebyshev truncations of $f(x) = \tanh(4x - 1)$ using the `'trunc'` flag in the Chebfun constructor. Another method is to compute all the Chebyshev coefficients of f and then truncate the series. Compute f_4 by this method and verify that the results agree to machine precision.

Exercise 4.7. When projection equals interpolation. Sometimes the projection f_n and the interpolant p_n are identical, even though both differ from f. Characterize exactly when this occurs, and give an example with $n = 3$.

5. Barycentric interpolation formula

How does one evaluate a Chebyshev interpolant? One good approach, involving $O(n \log n)$ work for a single point evaluation, is to compute Chebyshev coefficients and use the Chebyshev series. However, there is a direct method requiring just $O(n)$ work, not based on the series expansion, that is both elegant and numerically stable. It also has the advantage of generalizing to sets of points other than Chebyshev. It is called the *barycentric interpolation formula,* introduced by Salzer [1972], with an earlier closely related formula due to Marcel Riesz [1916]. The more general barycentric formula for arbitrary interpolation points, of which Salzer's formula is an exceptionally simple special case, was developed earlier by Dupuy [1948], with origins at least as early as Jacobi [1825]. Taylor [1945] introduced the barycentric formula for equispaced grid points. For a survey of barycentric formulas, see [Berrut & Trefethen 2004].

The study of polynomial interpolation goes back a long time; the word "interpolation" may be due to Wallis in 1656 (see [Pearson 1920] for an early account of some of the history.) In particular, Newton addressed the topic and devised a method based on divided differences. Many textbooks claim that it is important to use Newton's formulation for reasons of numerical stability, but this is not true, and we shall not discuss Newton's approach here.

Instead, the barycentric formula is of the alternative *Lagrange form,* where the interpolant is written as a linear combination of *Lagrange* or *cardinal* or *fundamental polynomials*:

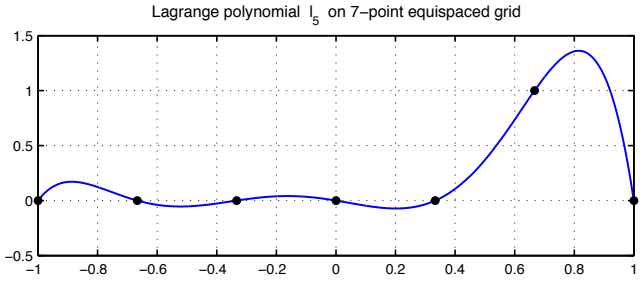
$$p(x) = \sum_{j=0}^{n} f_j \ell_j(x). \tag{5.1}$$

Here we have a set of distinct interpolation points $x_0, \ldots, x_n$, which could be real or complex, and $\ell_j(x)$, the jth Lagrange polynomial, is the unique polynomial in $\mathcal{P}_n$ that takes the value 1 at x_j and 0 at the other points x_k:

$$\ell_j(x_k) = \begin{cases} 1 & k = j, \\ 0 & k \neq j. \end{cases} \tag{5.2}$$

For example, here is a plot of ℓ_5 on the equispaced 7-point grid (i.e., $n = 6$):

```
d = domain(-1,1); s = linspace(-1,1,7); y = [0 0 0 0 0 1 0];
p = interp1(s,y,d); plot(p), hold on, plot(s,p(s),'.k')
```



It is easy to write down an explicit expression for ℓ_j:

$$\ell_j(x) = \frac{\prod_{k \neq j}(x - x_k)}{\prod_{k \neq j}(x_j - x_k)}. \tag{5.3}$$

Since the denominator is a constant, this function is a polynomial of degree n with zeros at the right places, and clearly it takes the value 1 when $x = x_j$. Equation (5.3) is very well known and can be found in many textbooks as a standard representation for Lagrange interpolants. Lagrange worked with (5.1) and (5.3) in 1795 [Lagrange 1795], and his name is firmly attached to these ideas,[2] but the same formulas were published earlier by Waring [1779] and Euler [1783], who had been Lagrange's predecessor at the Berlin Academy.

Computationally speaking, (5.1) is excellent but (5.3) is not so good. It requires $O(n)$ operations to evaluate $\ell_j(x)$ for each value of x, and then $O(n)$ such evaluations must be added up in (5.1), giving a total operation count of $O(n^2)$ for evaluating $p(x)$ at a single value of x.

By a little rearrangement we can improve the operation count. The key observation is that for the various values of j, the numerators in (5.3) are the same except that they are missing different factors $x - x_j$. To take advantage of this commonality, we define the *node polynomial* $\ell \in \mathcal{P}_{n+1}$ for the given grid by

$$\ell(x) = \prod_{k=0}^{n} (x - x_k). \tag{5.4}$$

[2] Perhaps Cauchy did some of the attaching, since he wrote in his *Cours d'analyse,* "Cette formule, donnée pour la première fois par Lagrange, . . ." [Cauchy 1821].

Then (5.3) becomes the elementary but extremely important identity

$$\ell_j(x) = \frac{\ell(x)}{\ell'(x_j)(x - x_j)}. \tag{5.5}$$

(We shall use this equation to derive the Hermite integral formula in Chapter 11.) Equivalently, let us define

$$\lambda_j = \frac{1}{\prod_{k \neq j}(x_j - x_k)}, \tag{5.6}$$

that is,

$$\lambda_j = \frac{1}{\ell'(x_j)}. \tag{5.7}$$

Then (5.3) becomes

$$\ell_j(x) = \ell(x)\frac{\lambda_j}{x - x_j}, \tag{5.8}$$

and the Lagrange formula (5.1) becomes

$$p(x) = \ell(x) \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j} f_j. \tag{5.9}$$

These formulas were derived by Jacobi in his PhD thesis in Berlin [Jacobi 1825], and they appeared in 19th century textbooks.[3]

Equation (5.9) has been called the "modified Lagrange formula" (by Higham) and the "first form of the barycentric interpolation formula" or the "type 1 barycentric formula" (starting with Rutishauser). What is valuable here is that the dependence on x inside the sum is so simple. If the weights $\{\lambda_j\}$ are known, (5.9) produces each value $p(x)$ with just $O(n)$ operations. Computing the weights from (5.6) requires $O(n^2)$ operations, but this computation only needs to be done once and for all, independently of x; and for special grids $\{x_j\}$ such as Chebyshev, as we shall see in a moment, the weights are known analytically and don't need to be computed at all. (For Legendre and other grids associated with orthogonal polynomials, the necessary computations can be carried out very fast; see Exercise 5.11 and Theorem 19.6.)

However, there is another barycentric formula that is more elegant. If we add up all the Lagrange polynomials ℓ_j, we get a polynomial in $\mathcal{P}_n$ that takes the value 1 at every point of the grid. Since polynomial interpolants are unique, this must be the constant polynomial 1:

$$\sum_{j=0}^{n} \ell_j(x) = 1.$$

[3] I am grateful to Folkmar Bornemann for drawing this history to my attention.

Dividing (5.8) by this expression enables us to cancel the factor $\ell(x)$, giving

$$\ell_j(x) = \frac{\lambda_j}{x - x_j} \Bigg/ \sum_{k=0}^{n} \frac{\lambda_k}{x - x_k}. \tag{5.10}$$

By inserting these representations in (5.1), we get the "second form of the barycentric interpolation formula" or "true barycentric formula" for polynomial interpolation in an arbitrary set of $n+1$ points $\{x_j\}$.

Theorem 5.1. Barycentric interpolation formula. *The polynomial interpolant through data $\{f_j\}$ at $n+1$ points $\{x_j\}$ is given by*

$$p(x) = \sum_{j=0}^{n} \frac{\lambda_j f_j}{x - x_j} \Bigg/ \sum_{j=0}^{n} \frac{\lambda_j}{x - x_j}, \tag{5.11}$$

with the special case $p(x) = f_j$ if $x = x_j$ for some j, where the weights $\{\lambda_j\}$ are defined by

$$\lambda_j = \frac{1}{\prod_{k \neq j}(x_j - x_k)}. \tag{5.12}$$

Proof. Given in the discussion above. ∎

It is obvious that the function defined by (5.11) interpolates the data. As x approaches one of the values x_j, one term in the numerator blows up and so does one term in the denominator. Their ratio is f_j, so this is clearly the value approached as x approaches x_j. On the other hand if x is equal to x_j, we can't use the formula: that would be a division of ∞ by ∞. This is why the theorem is stated with the qualification for the special case $x = x_j$.

What is not obvious is that the function defined by (5.11) is a polynomial, let alone a polynomial of degree n: it looks like a rational function. The fact that it is a polynomial depends on the special values (5.12) of the weights. For choices of nonzero weights that differ from (5.12), (5.11) will still interpolate the data, but in general it will be a rational function that is not a polynomial. These rational barycentric interpolants can be very useful in some applications, and they are likely to get more attention in the future [Berrut, Baltensperger & Mittelmann 2005, Tee & Trefethen 2006, Floater & Hormann 2007, Berrut, Floater & Klein 2011].

Chebfun's overload of the Matlab `interp1` command, which was illustrated at the beginning of this chapter, incorporates an implementation of (5.11)–(5.12). We shall make use of `interp1` again in Exercise 5.7 and in Chapters 13 and 15. Now, however, let us turn to the special case that is so important in practice.

For Chebyshev points, the weights $\{\lambda_j\}$ are wonderfully simple: they are equal to $(-1)^j$ times the constant $2^{n-1}/n$, or half this value for $j = 0$ and n. These numbers were worked out by Marcel Riesz in 1916 [Riesz 1916]. The constant cancels in the numerator and denominator when we divide by the

formula for 1 in (5.11), giving Salzer's amazingly simple result from 1972 [Salzer 1972]:

Theorem 5.2. Barycentric interpolation in Chebyshev points. *The polynomial interpolant through data $\{f_j\}$ at the Chebyshev points* (2.2) *is*

$$p(x) = \sum_{j=0}^{n}{}' \frac{(-1)^j f_j}{x - x_j} \Bigg/ \sum_{j=0}^{n}{}' \frac{(-1)^j}{x - x_j}, \tag{5.13}$$

with the special case $p(x) = f_j$ if $x = x_j$. The primes on the summation signs signify that the terms $j = 0$ and $j = n$ are multiplied by $1/2$.

Equation (5.13) is scale-invariant: for interpolation in Chebyshev points scaled to any interval $[a, b]$, the formula is exactly the same. This is a big advantage on the computer when n is in the thousands or higher, because it means that we need not worry about underflow or overflow.

Proof. Equation (5.13) is a special case of (5.11). To prove it, we will show that for Chebyshev points, the weights (5.12) reduce to $(-1)^j$ times the constant $2^{n-1}/n$, and half this value for $j = 0$ or n. To do this, we begin by noting that for Chebyshev points, the node polynomial (5.4) can be written as $\ell(x) = 2^{-n}(T_{n+1}(x) - T_{n-1}(x))$ (Exercise 4.1). Together with (5.8), this implies

$$\ell_j(x) = 2^{-n} \lambda_j \frac{T_{n+1}(x) - T_{n-1}(x)}{x - x_j},$$

and from (5.7) we have

$$\lambda_j = \frac{1}{\ell'(x_j)} = \frac{2^n}{T'_{n+1}(x_j) - T'_{n-1}(x_j)}.$$

Now it can be shown that

$$T'_{n+1}(x_j) - T'_{n-1}(x_j) = 2n(-1)^j, \quad 1 \le j \le n-1,$$

with twice this value for $j = 0$ and n (Exercise 5.3). So we have

$$\lambda_j = \frac{2^{n-1}}{n}(-1)^j, \quad 1 \le j \le n-1, \tag{5.14}$$

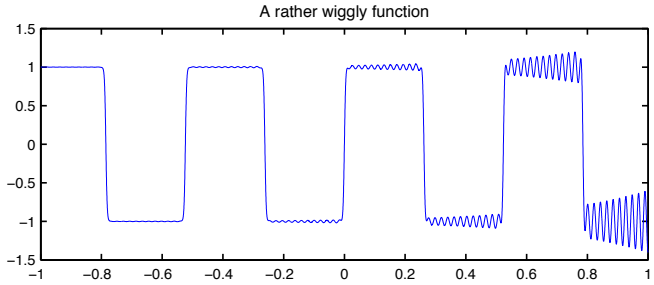
with half this value for $j = 0$ and n, as claimed. ∎

The formula (5.13) is extraordinarily effective, even if n is in the thousands or millions, even if p must be evaluated at thousands or millions of points. As a first example, let us construct a rather wiggly chebfun:

```
x = chebfun('x');
f = tanh(20*sin(12*x)) + .02*exp(3*x).*sin(300*x); length(f)

ans = 5158
```

We now plot `f` using 10000 sample points and note the time required:

```
tic, plot(f,'numpts',10000), toc

Elapsed time is 0.871149 seconds.
```



In this short time, Chebfun has evaluated a polynomial interpolant of degree about 5000 at 10000 sample points.

Raising the degree further, let p be the Chebyshev interpolant of degree 10^6 to the function $\sin(10^5 x)$ on $[-1,1]$:

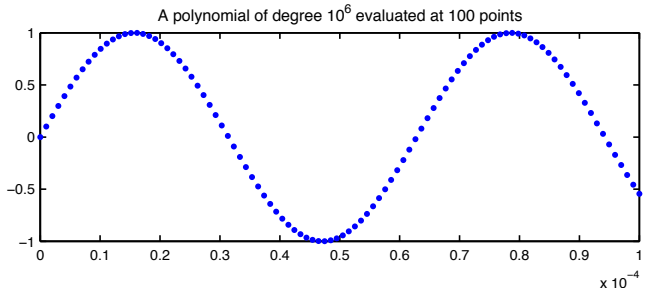
```
ff = @(x) sin(1e5*x); p = chebfun(ff,1000001);
```

How long does it take to evaluate this interpolant at 100 points?

```
xx = linspace(0,0.0001); tic, pp = p(xx); toc

Elapsed time is 1.224548 seconds.
```

Not bad for a million-degree polynomial! The result looks fine,

```
plot(xx,pp,'.')
```



and it matches the target function closely:

```
for j = 1:5, r = rand; disp([ff(r) p(r) ff(r)-p(r)]), end
```

```
    0.705930356624765   0.705930356617951   0.000000000006814
   -0.931512002954607  -0.931512002958003   0.000000000003395
    0.583585101736752   0.583585101743138  -0.000000000006386
   -0.851482366899905  -0.851482366903565   0.000000000003660
    0.988082673530624   0.988082673532397  -0.000000000001773
```

The apparent loss of 4 or 5 digits of accuracy is to be expected since the derivative of this function is of order 10^5: each evaluation is the correct result for a value of x within about 10^{-16} of the correct one (Exercise 5.5).

Experiments like these show that barycentric interpolation in Chebyshev points is a robust process: it is numerically stable, untroubled by rounding errors on a computer. This may seem surprising if you look at (5.9) or (5.13)—shouldn't cancellation errors on a computer cause trouble if x is close to one of the Chebyshev points x_j? In fact they do not, and these formulas have been proved stable in floating point arithmetic for all $x \in [-1,1]$ [Rack & Reimer 1982, Higham 2004]. This is in marked contrast to the more familiar algorithm of polynomial interpolation via solution of a Vandermonde linear system of equations, which is exponentially unstable (Exercise 5.2).

We must emphasize that whereas (5.13) is stable for interpolation, it is unstable for extrapolation, that is, the evaluation of $p(x)$ for $x \notin [-1,1]$. The more general formula (5.11) is unstable for extrapolation too and is unstable even for interpolation when used with arbitrary points rather than points suitably clustered like Chebyshev points. In these cases it is important to use the "type 1" barycentric formula (5.9) instead, which Higham proved stable in all cases. The disadvantage of (5.9) is that when n is larger than about a thousand, it is susceptible to troubles of underflow or overflow, which must be countered by rescaling $[-1,1]$ to $[-2,2]$ or by computing products by addition of logarithms.

More precisely, Higham [2004] showed that when they are used to evaluate $p(x)$ for $x \in [-1,1]$ with data at Chebyshev points, both (5.9) and (5.11)–(5.13) have a certain property that numerical analysts call *forward stability*. If you want to evaluate $p(x)$ for values of x outside $[-1,1]$, however, (5.11)–(5.13) lose their stability and it is important to use (5.9), which has the stronger property known as *backward stability* [Webb, Trefethen & Gonnet 2011]. It is also important to use (5.9) rather than (5.11) for computing interpolants through equispaced points or other point sets that are far from the Chebyshev distribution. (As we shall discuss in Chapters 13–14, in these cases the problem is probably so ill-conditioned that one should not be doing polynomial interpolation in the first place.)

These observations show that (5.9) has advantages over (5.11) and (5.13), but it also has an important disadvantage: it is not scale-invariant, and the weights grow exponentially as functions of the inverse of the length of the interval of interpolation. We see this in (5.14), where the weights have size 2^n, and would in fact overflow on a computer in standard IEEE double precision arithmetic for n bigger than about 1000. (Higham's analysis ignores overflow and underflow.)

We shall have more to say about this exponential dependence in Chapters 11–15. So (5.11) and (5.13) remain a good choice for most applications, so long as the interpolation points are Chebyshev or similar and the evaluation points lie in $[-1,1]$.

SUMMARY OF CHAPTER 5. *Polynomial interpolants can be evaluated fast and stably by the barycentric formula, even for thousands or millions of interpolation points. The barycentric formula has the form of a rational function, but reduces to a polynomial because of the use of specially determined weights.*

Exercise 5.1. Barycentric coefficients by hand. (a) Work out on paper the barycentric interpolation coefficients $\{\lambda_j\}$ for the case $n = 3$ and $x_0 = -1$, $x_1 = 0$, $x_2 = 1/2$, $x_3 = 1$. (b) Confirm that (5.9) gives the right value $p(-1/2)$ for the polynomial interpolant to data $1, 2, 3, 4$ in these points.

Exercise 5.2. Instability of Vandermonde interpolation. The best-known numerical algorithm for polynomial interpolation, unlike the barycentric formula, is unstable. This is the method implemented in the Matlab `polyfit` command, which forms a Vandermonde matrix of sampled powers of x and solves a corresponding linear system of equations. (In [Trefethen 2000], to my embarrassment, this unstable method is used throughout, forcing the values of n used for plots in that book to be kept small.) (a) Explore this instability by comparing a Chebfun evaluation of $p(0)$ with the result of `polyval(polyfit(xx,f(xx),n),0)` where `f = @(x) cos(k*x)` for $k = 10, 20, \ldots, 90, 100$, n is the degree of the corresponding chebfun, and `xx` is a fine grid. (b) Examining the Matlab `polyfit` code as appropriate, construct the Vandermonde matrices V for each of these ten problems and compute their condition numbers. (You can also use the Matlab `vander` command.) By contrast, the underlying Chebyshev interpolation problem is well-conditioned.

Exercise 5.3. Calculating derivatives for the proof of Theorem 5.2. Derive the following identities used in the proof of Theorem 5.2. (a) For $1 \le j \le n-1$, $T'_{n+1}(x_j) - T'_{n-1}(x_j) = 2n(-1)^j$. (b) For $j = 0$ and $j = n$, $T'_{n+1}(x_j) - T'_{n-1}(x_j) = 4n(-1)^j$. One can derive this formula directly, or indirectly by a symmetry argument.

Exercise 5.4. Interpolating the sign function. Use `x = chebfun('x')`, `f = sign(x)` to construct the sign function on $[-1,1]$ and `p = chebfun('sign(x)',10000)` to construct its interpolant in 10000 Chebyshev points. Explore the difference in the interesting region by defining `d = f-p`, `d = d{-0.002,0.002}`. What is the maximum value of `p`? In what subset of $[-1,1]$ is `p` smaller than 0.5 in absolute value?

Exercise 5.5. Accuracy of point evaluations. (a) Construct the chebfun g corresponding to $f(x) = \sin(\exp(10x))$ on $[-1,1]$. What is the degree of this polynomial? (b) Let `xx` be the vector of 1000 linearly spaced points from -1 to 1. How long does it take on your computer to evaluate f(`xx`)? g(`xx`)? (c) Draw a loglog plot of the vector of errors $|f(\texttt{xx}) - g(\texttt{xx})|$ against the vector of derivatives $|f'(\texttt{xx})|$. Comment on why the dots line up as they do.

Exercise 5.6. Equispaced points. Show that for equispaced points in $[-1,1]$ with spacing h, the barycentric weights are $\lambda_j = (-1)^{n-j}/(j!(n-j)!h^n)$, or after canceling common factors, $\lambda_j = (-1)^j \binom{n}{j}$ [Taylor 1945].

Exercise 5.7. A greedy algorithm for choosing interpolation grids. Write a program using Chebfun's `interp1` command to compute a sequence of polynomial

interpolants to a function f on $[-1,1]$ in points selected by a greedy algorithm: take x_0 to be a point where $|f(x)|$ achieves its maximum, then x_1 to be a point where $|(f-p_0)(x)|$ achieves its maximum, then x_2 to be a point where $|(f-p_1)(x)|$ achieves its maximum, and so on. Plot the error curves $(f-p_n)(x)$, $x \in [-1,1]$ computed by this algorithm for $f(x) = |x|$ and $0 \le n \le 25$. Comment on the spacing of the grid $\{x_0, \dots, x_{25}\}$.

Exercise 5.8. Barycentric formula for Chebyshev polynomials. Derive an elegant formula for $T_n(x)$ from (5.13) [Salzer 1972].

Exercise 5.9. Barycentric interpolation in roots of unity. Derive the barycentric weights $\{\lambda_j\}$ for polynomial interpolation in (a) $\{\pm 1\}$, (b) $\{1, i, -1, -i\}$, (c) The $(n+1)$st roots of unity for arbitrary $n \ge 0$.

Exercise 5.10. Barycentric weights for a general interval. (a) How does the formula (5.14) for Chebyshev barycentric weights on $[-1,1]$ change for weights on an interval $[a,b]$? (b) The *capacity* of $[a,b]$ (see Chapter 12) is equal to $c = (b-a)/4$. How do the barycentric weights behave as $n \to \infty$ for an interval of capacity c? As a function of c, what is the maximal value of n for which they can be represented in IEEE double precision arithmetic without overflow or underflow? (You may assume the overflow and underflow limits are 10^{308} and 10^{-308}. The overflow/underflow problem goes away with the use of the divided form (5.13).)

Exercise 5.11. Barycentric interpolation in Legendre points. Chebfun includes fast algorithms for computing barycentric weights for various distributions of points other than Chebyshev, such as Legendre points, the zeros of Legendre polynomials (see Chapter 17 and Theorem 19.6). Perform a numerical experiment to compare the accuracy of interpolants in Chebyshev and Legendre points to $f(x) = e^x \sin(300x)$ at $x = 0.99$. Specifically, compute `[s,w,lambda] = legpts(n+1)` and `bary(0.99,f(s),s,lambda)` for $1 \le n \le 500$ and make a semilog plot of the absolute value of the error as a function of n; compare this with the analogous plot for Chebyshev points.

Exercise 5.12. Barycentric rational interpolation. (a) If the formula (5.13) is used with points $\{x_j\}$ other than Chebyshev with maximum spacing h, it produces a rational interpolant of accuracy $O(h^2)$ as $h \to 0$ [Berrut 1988]. Confirm this numerically for $f(x) = e^x$ and equispaced points in $[-1,1]$. (b) Show numerically that the accuracy improves to $O(h^3)$ if the pattern of coefficients near the left end is changed from $\frac{1}{2}, -1, 1, -1, \dots$ to $\frac{1}{4}, -\frac{3}{4}, 1, -1, \dots$ and analogously at the right end [Floater & Hormann 2007].

Exercise 5.13. Barycentric weights and geometric mean distances. (a) Give an interpretation of (5.6) in terms of geometric mean distances between grid points. (b) Explain how one of the theorems of this chapter explains the result of Exercise 2.6.

6. Weierstrass approximation theorem

Every continuous function on a bounded interval can be approximated to arbitrary accuracy by polynomials. This is the famous Weierstrass approximation theorem, proved by Karl Weierstrass when he was 70 years old [Weierstrass 1885]. The theorem was independently discovered at about the same time, in essence, by Carl Runge: as pointed out in 1886 by Phragmén in remarks published as a footnote stretching over four pages in a paper by Mittag-Leffler [1900], it can be derived as a corollary of results Runge published in a pair of papers in 1885 [Runge 1885A & 1885B].

Here and throughout this book, unless indicated otherwise, $\|\cdot\|$ denotes the supremum norm on $[-1,1]$.

Theorem 6.1. Weierstrass approximation theorem. *Let f be a continuous function on $[-1,1]$, and let $\varepsilon > 0$ be arbitrary. Then there exists a polynomial p such that*

$$\|f - p\| < \varepsilon.$$

Outline of proof. We shall not spell out an argument in detail. However, here is an outline of the beautiful proof from Weierstrass's original paper. First, extend $f(x)$ to a continuous function $\tilde{f}$ with compact support on the whole real line. Now, take $\tilde{f}$ as initial data at $t = 0$ for the diffusion equation $\partial u/\partial t = \partial^2 u/\partial x^2$ on the real line. It is known that by convolving $\tilde{f}$ with the Gaussian kernel $\phi(x) = e^{-x^2/4t}/\sqrt{4\pi t}$, we get a solution to this partial differential equation that converges uniformly to f as $t \to 0$, and thus can be made arbitrarily close to f on $[-1,1]$ by taking t small enough. On the other hand, since $\tilde{f}$ has compact support, for each $t > 0$ this solution is an integral

over a bounded interval of entire functions and is thus itself an entire function, that is, analytic throughout the complex plane. Therefore it has a uniformly convergent Taylor series on $[-1,1]$, which can be truncated to give polynomial approximations of arbitrary accuracy. ∎

For a fuller presentation of the argument just given as "one of the most amusing applications of the Gaussian kernel," where the result is stated for the more general case of a function of several variables approximated by multivariate polynomials, see [Folland 1995].

Many other proofs of the Weierstrass theorem are also known, including these early ones:

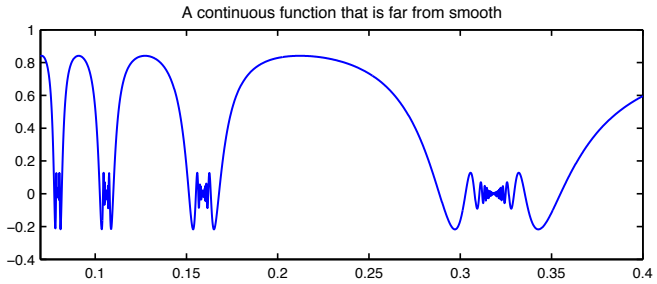
Runge (1885)
Picard (1891)
Lerch (1892 and 1903)
Volterra (1897)
Lebesgue (1898)
Mittag-Leffler (1900)
Fejér (1900 and 1916)
Landau (1908)
de la Vallée Poussin (1908)
Jackson (1911)
Sierpinski (1911)
Bernstein (1912)
Montel (1918)

For example, Bernstein's proof is a discrete analogue of the argument just given: continuous diffusion is replaced by a random walk made precise by the notion of *Bernstein polynomials* (Exercise 6.4) [Bernstein 1912D]. Lebesgue's proof, which appeared in his first paper published as a student at age 23, is based on reducing the approximation of general continuous functions to the approximation of $|x|$ (Exercise 6.5) [Lebesgue 1898]. Fejér was an even younger student, age 20, when he published his proof based on *Cesàro means* (Exercise 6.6a) [Fejér 1900], and he published a different proof years later based on *Hermite–Fejér interpolation* (Exercise 6.6b) [Fejér 1916]. This long list gives an idea of the great amount of mathematics stimulated by Weierstrass's theorem and the significant role it played in the development of analysis in the early 20th century. For a fascinating presentation of this corner of mathematical history, see [Pinkus 2000].

Weierstrass's theorem establishes that even extremely non-smooth functions can be approximated by polynomials, functions like $x\sin(x^{-1})$ or even $\sin(x^{-1})\sin(1/\sin(x^{-1}))$. The latter function has an infinite number of points near which it oscillates infinitely often, as we begin to see from the plot below over the range $[0.07, 0.4]$. In this calculation Chebfun is called with a user-prescribed number of interpolation points, 30,000, since the usual adaptive procedure has no chance of resolving the function to machine precision.

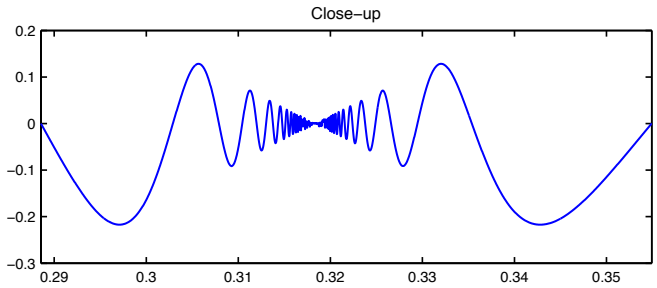
```
f = chebfun(@(x) sin(1./x).*sin(1./sin(1./x)),[.07 .4],30000);
```

```
plot(f)
```



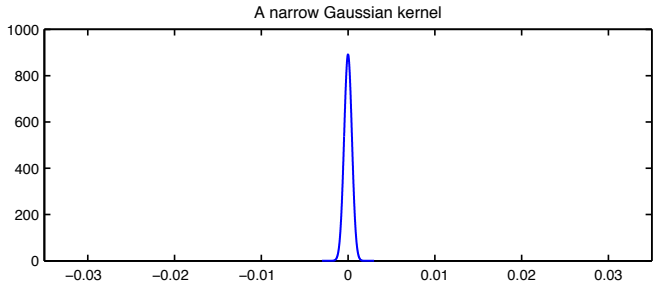
We can illustrate the idea of Weierstrass's proof by showing the convolution of this complicated function with a Gaussian. First, here is the same function f recomputed over a subinterval extending from one of its zeros to another:

```
a = 0.2885554757; b = 0.3549060246;
f2 = chebfun(@(x) sin(1./x).*sin(1./sin(1./x)),[a,b],2000);
plot(f2)
```



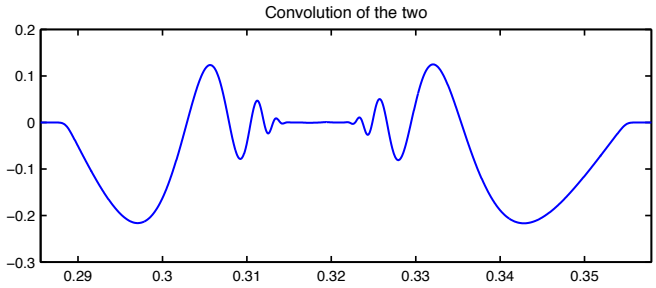
Here is a narrow Gaussian with integral 1.

```
t = 1e-7;
phi = chebfun(@(x) exp(-x.^2/(4*t))/sqrt(4*pi*t),.003*[-1 1]);
plot(phi)
```



Convolving the two gives a smoothed version of the close-up of f. Notice how the short wavelengths vanish while the long ones are nearly undisturbed.

```
f3 = conv(f2,phi); plot(f3)
```



This is an entire function, which means it can be approximated by polynomials by truncating the Taylor series.

Weierstrass's theorem has an important generalization to complex analytic functions. Suppose a function f is defined on a compact set K in the complex plane whose complement is connected (so K cannot have any holes). *Mergelyan's theorem* asserts that if f is continuous on K and analytic in the interior, then f can be approximated on K by polynomials [Mergelyan 1951, Gaier 1987]. The earlier *Runge's theorem* is the weaker result in which f is asumed to be analytic throughout K, not just in the interior [Runge 1885a].

For all its beauty, power, and importance, the Weierstrass approximation theorem has in some respects served as an unfortunate distraction. Knowing that even troublesome functions can be approximated by polynomials, we naturally ask, how can we do it? A famous result of Faber and Bernstein asserts that there is no fixed array of grids of $1, 2, 3, \ldots$ interpolation points, Chebyshev or otherwise, that achieves convergence as $n \to \infty$ for all continuous f [Faber 1914, Bernstein 1919]. So it becomes tempting to look at approximation methods that go beyond interpolation, and to warn people that interpolation is dangerous, and to try to characterize exactly what minimal properties of f suffice to ensure that interpolation will work after all. A great deal is known about these subjects. The trouble with this line of research is that for almost all the functions encountered in practice, Chebyshev interpolation works beautifully! Weierstrass's theorem has encouraged mathematicians over the years to give too much of their attention to pathological functions at the edge of discontinuity, leading to the bizarre and unfortunate situation where many books on numerical analysis caution their readers that interpolation may fail without mentioning that for functions with a little bit of smoothness, it succeeds outstandingly. For a discussion of the history of such misrepresentations and misconceptions, see Chapter 14 and also the appendix on "Six myths of polynomial interpolation and quadrature."

Summary of Chapter 6. *A continuous function on a bounded interval can be approximated arbitrarily closely by polynomials.*

Exercise 6.1. A pathological function of Weierstrass. Weierstrass was one of the first to give an example of a function continuous but nowhere differentiable on

$[-1,1]$, and it is one of the early examples of a fractal [Weierstrass 1872]:

$$w(x) = \sum_{k=0}^{\infty} 2^{-k} \cos(3^k x). \tag{6.1}$$

(a) Construct a chebfun `w7` corresponding to this series truncated at $k = 7$. Plot `w7`, its derivative (use `diff`), and its indefinite integral (`cumsum`). What is the degree of the polynomial defining this chebfun? (b) Prove that w is continuous. (You can use the Weierstrass M-test.)

Exercise 6.2. Taylor series of an entire function. To illustrate the proof of the Weierstrass approximation theorem, we plotted a Gaussian kernel. The key point of the proof is that this kernel is entire, so its Taylor series converges for all x. (a) For $x = 1$ at the given time $t = 10^{-7}$, how many terms of the Taylor series about $x = 0$ would you have to take before the terms fall below 1? Estimate the answer at least to within a factor of 2. You may find Stirling's formula helpful. (b) Also for $x = 1$ and $t = 10^{-7}$, approximately how big is the biggest term in the Taylor series?

Exercise 6.3. Resolving a difficult function. Although the example function $f(x) = \sin(1/x)\sin(1/\sin(1/x))$ of this chapter is not Lipschitz continuous, its Chebyshev interpolants do in fact converge. Explore this phenomenon numerically by computing the degree n Chebyshev interpolant to f over the interval $[0.07, 0.4]$ for $n+1 = 4, 8, 16, \ldots, 2^{14}$ and measuring the error in each case over a Chebyshev grid of $2n$ points. Plot the results on a loglog scale. How do you think the error depends on n as $n \to \infty$? Approximately how large would n have to be to get 16-digit accuracy for this function over this interval?

Exercise 6.4. Bernstein's proof. For $f \in C([0,1])$, the associated degree n *Bernstein polynomial* is defined by

$$B_n(x) = \sum_{k=0}^{n} f(k/n) \binom{n}{k} x^k (1-x)^{n-k}. \tag{6.2}$$

Bernstein proved the Weierstrass approximation theorem by showing that $B_n(x) \to f(x)$ uniformly as $n \to \infty$. (a) Give an interpretation of $B_n(x)$ involving a random walk driven by a coin which comes up heads with probability x and tails with probability $1-x$. (b) Show that $\max B_n(x) \le \max f(x)$ and $\min B_n(x) \ge \min f(x)$ for $x \in [0,1]$.

Exercise 6.5. Lebesgue's proof. (a) Show using uniform continuity that any $f \in C([-1,1])$ can be approximated uniformly by a polygonal curve, i.e., a function $g(x)$ that is piecewise linear and continuous. (b) Show that such a function can be written in the form $g(x) = A + Bx + \sum_{k=1}^{m} C_k |x - x_k|$. (c) Show that $|x|$ can be uniformly approximated by polynomials on $[-1,1]$ by truncating the binomial expansion

$$[1 - (1-x^2)]^{1/2} = \sum_{k=0}^{\infty} \binom{\frac{1}{2}}{n} (x^2 - 1)^n.$$

You may use without proof the fact that these binomial coefficients are of size $O(n^{-3/2})$ as $n \to \infty$. (d) Explain how (a)–(c) combine to give a proof of the Weierstrass approximation theorem.

Exercise 6.6. Fejér's proofs. (a) In 1900 Fejér proved the Weierstrass approximation theorem via *Cesàro means*. In the Chebyshev case, define S_n to be the mean of the partial sums of the Chebyshev series (3.11)–(3.12) of orders 0 through n. Then

it can be shown that $S_n \to f$ uniformly as $n \to \infty$ for any $f \in C([-1,1])$. Explore such approximations for $f(x) = e^x$ with various degrees n. For this very smooth function f, how does the accuracy compare with that of ordinary Chebyshev interpolants? (b) In 1916 Fejér proved the theorem again by considering what are now known as *Hermite–Fejér interpolants*: he showed that if $p_{2n} \in \mathcal{P}_{2n}$ is obtained by interpolating $f \in C([-1,1])$ in the zeros of $T_n(x)$ and also setting $p'(x) = 0$ at these points, then $p_{2n} \to f$ uniformly as $n \to \infty$. Explore such interpolants numerically for various n by using `interp1` to construct polynomials p_{2n} with $p_{2n}(x_j) = p_{2n}(x_j + 10^{-6}) = \exp(x_j)$. Again how does the accuracy compare with that of ordinary Chebyshev interpolants?

Exercise 6.7. Convergent series of polynomials. (a) Show that any $f \in C([-1,1])$ can be written as a uniformly convergent series

$$f(x) = \sum_{k=0}^{\infty} q_k(x),$$

where each q_k is a polynomial of some degree. (b) Show that a series of the same kind also exists for a function continuous on the whole real line, with pointwise convergence for all x and uniform convergence on any bounded subset.