# Spectral Methods in MATLAB

Lloyd N. Trefethen

Download the programs from
`http://www.comlab.ox.ac.uk/oucl/work/nick.trefethen.`

Start up MATLAB.

Run `p1`, `p2`, `p3`, . . . .

Happy computing!

| **TOPIC** | **PROGRAMS** |
|---|---|
| Chebyshev differentiation matrices | 11, 26 |
| Chebyshev differentiation by FFT | 18, 19, 20 |
| Clenshaw–Curtis quadrature | 30 |
| Complex arithmetic | 10, 24, 25, 31, 40 |
| Differentiation | 1, 2, 4, 5, 7, 11, 12, 18 |
| Eigenvalue problems | 8, 15, 21, 22, 23, 24, 26, 28, 39, 40 |
| Finite difference methods | 1 |
| Fourier differentiation matrices | 2, 4, 8 |
| Fourier differentiation by FFT | 5, 6, 28 |
| Fourth-order problems | 38, 39, 40 |
| Gauss quadrature | 30 |
| Gibbs phenomenon | 3 |
| Inhomogeneous boundary conditions | 32, 35, 36 |
| Interpolation | 3, 9, 10, 13, 16 |
| Laplace and Poisson problems | 13, 14, 15, 16, 23, 28, 29, 32, 33, 36 |
| Neumann boundary conditions | 33, 37 |
| Nonlinear problems | 14, 27, 34, 35 |
| Periodic domains | 4, 21 |
| Polar coordinates | 28, 29 |
| Potential theory | 10 |
| Pseudospectra | 24 |
| Runge phenomenon | 9, 10 |
| Spectral accuracy | 2, 4, 7, 8, 11, 12, 30 |
| Time-stepping | 6, 19, 20, 25, 27, 34, 35, 37 |
| Two-dimensional domains | 16, 17, 20, 23, 28, 29, 36, 37, 39 |
| Two-point boundary value problems | 13, 14, 32, 33, 38 |
| Unbounded domains | 8, 24 |
| Variable coefficients | 6, 8, 22, 23, 24 |
| Wave equations | 6, 19, 20, 27, 37 |

To Anne

# Contents

# Preface

The aim of this book is to teach you the essentials of spectral collocation methods with the aid of 40 short MATLAB® programs, or "M-files."* The programs are available online at `http://www.comlab.ox.ac.uk/oucl/work/nick.trefethen`, and you will run them and modify them to solve all kinds of ordinary and partial differential equations (ODEs and PDEs) connected with problems in fluid mechanics, quantum mechanics, vibrations, linear and nonlinear waves, complex analysis, and other fields. Concerning prerequisites, it is assumed that the words just written have meaning for you, that you have some knowledge of numerical methods, and that you already know MATLAB.

If you like computing and numerical mathematics, you will enjoy working through this book, whether alone or in the classroom—and if you learn a few new tricks of MATLAB along the way, that's OK too!

Spectral methods are one of the "big three" technologies for the numerical solution of PDEs, which came into their own roughly in successive decades:

1950s: finite difference methods
1960s: finite element methods
1970s: spectral methods

Naturally, the origins of each technology can be traced further back. For spectral methods, some of the ideas are as old as interpolation and expan-

*MATLAB is a registered trademark of The MathWorks, Inc., 3 Apple Hill Drive, Natick, MA 01760-2098, USA, tel. 508-647-7000, fax 508-647-7001, `info@mathworks.com`, `http://www.mathworks.com`.

sion, and more specifically algorithmic developments arrived with Lanczos as early as 1938 [Lan38, Lan56] and with Clenshaw, Elliott, Fox, and others in the 1960s [FoPa68]. Then, in the 1970s, a transformation of the field was initiated by work by Orszag and others on problems in fluid dynamics and meteorology, and spectral methods became famous. Three landmarks of the early modern spectral methods literature were the short book by Gottlieb and Orszag [GoOr77], the survey by Gottlieb, Hussaini, and Orszag [GHO84], and the monograph by Canuto, Hussaini, Quarteroni, and Zang [CHQZ88]. Other books have been contributed since then by Mercier [Mer89], Boyd [Boy00] (first edition in 1989), Funaro [Fun92], Bernardi and Maday [BeMa92], Fornberg [For96], and Karniadakis and Sherwin [KaSh99].

If one wants to solve an ODE or PDE to high accuracy on a simple domain, and if the data defining the problem are smooth, then spectral methods are usually the best tool. They can often achieve ten digits of accuracy where a finite difference or finite element method would get two or three. At lower accuracies, they demand less computer memory than the alternatives.

This short textbook presents some of the fundamental ideas and techniques of spectral methods. It is aimed at anyone who has finished a numerical analysis course and is familiar with the basics of applied ODEs and PDEs. You will see that a remarkable range of problems can be solved to high precision by a few lines of MATLAB in a few seconds of computer time. Play with the programs; make them your own! The exercises at the end of each chapter will help get you started.

I would like to highlight three mathematical topics presented here that, while known to experts, are not usually found in textbooks. The first, in Chapter 4, is the connection between the smoothness of a function and the rate of decay of its Fourier transform, which determines the size of the aliasing errors introduced by discretization; these connections explain how the accuracy of spectral methods depends on the smoothness of the functions being approximated. The second, in Chapter 5, is the analogy between roots of polynomials and electric point charges in the plane, which leads to an explanation in terms of potential theory of why grids for nonperiodic spectral methods need to be clustered at boundaries. The third, in Chapter 8, is the three-way link between Chebyshev series on $[-1, 1]$, trigonometric series on $[-\pi, \pi]$, and Laurent series on the unit circle, which forms the basis of the technique of computing Chebyshev spectral derivatives via the fast Fourier transform. All three of these topics are beautiful mathematical subjects in their own right, well worth learning for any applied mathematician.

If you are determined to move immediately to applications without paying too much attention to the underlying mathematics, you may wish to turn directly to Chapter 6. Most of the applications appear in Chapters 7–14.

Inevitably, this book covers only a part of the subject of spectral methods. It emphasizes collocation ("pseudospectral") methods on periodic and on

Chebyshev grids, saying next to nothing about the equally important Galerkin methods and Legendre grids and polynomials. The theoretical analysis is very limited, and simple tools for simple geometries are emphasized rather than the "industrial strength" methods of spectral elements and *hp* finite elements. Some indications of omitted topics and other points of view are given in the Afterword.

A new era in scientific computing has been ushered in by the development of MATLAB. One can now present advanced numerical algorithms and solutions of nontrivial problems in complete detail with great brevity, covering more applied mathematics in a few pages than would have been imaginable a few years ago. By sacrificing sometimes (not always!) a certain factor in machine efficiency compared with lower level languages such as Fortran or C, one obtains with MATLAB a remarkable human efficiency—an ability to modify a program and try something new, then something new again, with unprecedented ease. This short book is offered as an encouragement to students, scientists, and engineers to become skilled at this new kind of computing.

# Acknowledgments

I must begin by acknowledging two special colleagues who have taught me a great deal about spectral methods over the years. These are André Weideman, of the University of Stellenbosch, coauthor of the "MATLAB Differentiation Matrix Suite" [WeRe00], and Bengt Fornberg, of the University of Colorado, author of *A Practical Guide to Pseudospectral Methods* [For96]. These good friends share my enthusiasm for simplicity—and my enjoyment of the occasional detail that refuses to be simplified, no matter how hard you try. In this book, among many other contributions, Weideman significantly improved the crucial program `cheb`.

I must also thank Cleve Moler, the inventor of MATLAB, a friend and mentor since my graduate school days. Perhaps I had better admit here at the outset that there is a brass plaque on my office wall, given to me in 1998 by The MathWorks, Inc., which reads: *FIRST ORDER FOR MATLAB, February 7, 1985, Ordered by Professor Nick Trefethen, Massachusetts Institute of Technology.* I was there in the classroom at Stanford when Moler taught the numerical eigensystems course CS238b in the winter of 1979 based around this new experimental interface to EISPACK and LINPACK he had cooked up. I am a card-carrying MATLAB fan.

Toby Driscoll, author of the SC Toolbox for Schwarz–Christoffel conformal mapping in MATLAB [Dri96], has taught me many MATLAB tricks, and he helped to improve the codes in this book. He also provided key suggestions for the nonlinear waves calculations of Chapter 10. The other person whose suggestions improved the codes most significantly is Pascal Gahinet of The MathWorks, Inc., whose eye for MATLAB style is something special. David Carlisle

of NAG, Ltd., one of the authors of LaTeX $2_\varepsilon$, showed me how to make blank lines in MATLAB programs come out a little bit shortened when included as verbatim input, saving precious centimeters for display of figures. Walter Gautschi and Sotiris Notaris informed me about matters related to Clenshaw–Curtis quadrature, and Jean-Paul Berrut and Richard Baltensperger taught me about rounding errors in spectral differentiation matrices.

A number of other colleagues commented upon drafts of the book and improved it. I am especially grateful to John Boyd, Frédéric Dias, Des Higham, Nick Higham, Álvaro Meseguer, Paul Milewski, Damian Packer, and Satish Reddy.

In a category by himself goes Mark Embree, who has read this book more carefully than anyone else but me, by far. Embree suggested many improvements in the text, and beyond that, he worked many of the exercises, catching errors and contributing new exercises of his own. I am lucky to have found Embree at a stage of his career when he still has so much time to give to others.

The Numerical Analysis Group here at Oxford provides a stimulating environment to support a project like this. I want particularly to thank my three close colleagues Mike Giles, Endre Süli, and Andy Wathen, whose friendship has made me glad I came to Oxford; Shirley Dickson, who cheerfully made multiple copies of drafts of the text half a dozen times on short notice; and our outstanding group secretary and administrator, Shirley Day, who will forgive me, I hope, for all the mornings I spent working on this book when I should have been doing other things.

This book started out as a joint production with Andrew Spratley, a D. Phil. student, based on a masters-level course I taught in 1998 and 1999. I want to thank Spratley for writing the first draft of many of these pages and for major contributions to the book's layout and figures. Without his impetus, the book would not have been written.

Once we knew it would be written, there was no doubt who the publisher should be. It was a pleasure to publish my previous book [TrBa97] with SIAM, an organization that manages to combine the highest professional standards with a personal touch. And there was no doubt who the copy editor should be: again the remarkable Beth Gallagher, whose eagle eye and good sense have improved the presentation from beginning to end.

Finally, special thanks for their encouragement must go to my two favorite younger mathematicians, Emma (8) and Jacob (6) Trefethen, who know how I love differential equations, MATLAB, and writing. I'm the sort of writer who polishes successive drafts endlessly, and the children are used to seeing me sit down and cover a chapter with marks in red pen. Jacob likes to tease me and ask, "Did you find more mistakes in your book, Daddy?"

# A Note on the MATLAB Programs

The MATLAB programs in this book are terse. I have tried to make each one compact enough to fit on a single page, and most often, on half a page. Of course, there is a message in this style, which is the message of this book: you can do an astonishing amount of serious computing in a few inches of computer code! And there is another message, too. The best discipline for making sure you understand something is to simplify it, simplify it relentlessly.

Without a doubt, readability is sometimes impaired by this obsession with compactness. For example, I have often combined two or three short MATLAB commands on a single program line. You may prefer a looser style, and that is fine. What's best for a printed book is not necessarily what's best for one's personal work.

Another idiosyncrasy of the programming style in this book is that the structure is flat: with the exception of the function `cheb`, defined in Chapter 6 and used repeatedly thereafter, I make almost no use of functions. (Three further functions, `chebfft`, `clencurt`, and `gauss`, are introduced in Chapters 8 and 12, but each is used just locally.) This style has the virtue of emphasizing how much can be achieved compactly, but as a general rule, MATLAB programmers should make regular use of functions.

Quite a bit might have been written to explain the details of each program, for there are tricks throughout this book that will be unfamiliar to some readers. To keep the discussion focused on spectral methods, I made a deliberate decision not to mention these MATLAB details except in a very few cases. This means that as you work with the book, you will have to study the programs, not just read them. What is this "`pol2cart`" command in Program 28

(p. 120)? What's going on with the index variable "`b`" in Program 36 (p. 142)? You will only understand the answers to questions like these after you have spent time with the codes and adapted them to solve your own problems. I think this is part of the fun of using this book, and I hope you agree.

The programs listed in these pages were included as M-files directly into the LaTeX source file, so all should run correctly as shown. The outputs displayed are exactly those produced by running the programs on my machine. There was a decision involved here. Did we really want to clutter the text with endless formatting and Handle Graphics commands such as `fontsize`, `markersize`, `subplot`, and `pbaspect`, which have nothing to do with the mathematics? In the end I decided that yes, we did. I want you to be able to download these programs and get beautiful results immediately. Equally important, experience has shown me that the formatting and graphics details of MATLAB are areas of this language where many users are particularly grateful for some help.

My personal MATLAB setup is nonstandard in one way: I have a file `startup.m` that contains the lines

```
set(0,'defaultaxesfontsize',12,'defaultaxeslinewidth',.7,...
     'defaultlinelinewidth',.8,'defaultpatchlinewidth',.7).
```

This makes text appear by default slightly larger than it otherwise would, and lines slightly thicker. The latter is important in preparing attractive output for a publisher's high-resolution printer.

The programs in this book were prepared using MATLAB versions 5.3 and 6.0. As later versions are released in upcoming years, unfortunately, it is possible that some difficulties with the programs will appear. Updated codes with appropriate modifications will be made available online as necessary.

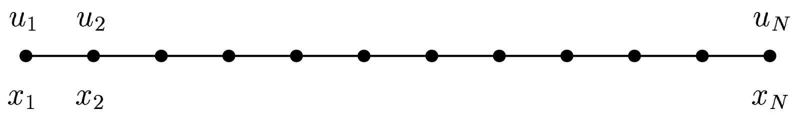
To learn MATLAB from scratch, or for an outstanding reference, I recommend SIAM's new *MATLAB Guide*, by Higham and Higham [HiHi00].

Think globally. Act locally.

# 1. Differentiation Matrices

Our starting point is a basic question. Given a set of grid points $\{x_j\}$ and corresponding function values $\{u(x_j)\}$, how can we use this data to approximate the derivative of $u$? Probably the method that immediately springs to mind is some kind of finite difference formula. It is through finite differences that we shall motivate spectral methods.

To be specific, consider a uniform grid $\{x_1, \ldots, x_N\}$, with $x_{j+1} - x_j = h$ for each $j$, and a set of corresponding data values $\{u_1, \ldots, u_N\}$:



Let $w_j$ denote the approximation to $u'(x_j)$, the derivative of $u$ at $x_j$. The standard second-order finite difference approximation is

$$w_j = \frac{u_{j+1} - u_{j-1}}{2h}, \tag{1.1}$$

which can be derived by considering the Taylor expansions of $u(x_{j+1})$ and $u(x_{j-1})$. For simplicity, let us assume that the problem is periodic and take $u_0 = u_N$ and $u_1 = u_{N+1}$. Then we can represent the discrete differentiation process as a matrix-vector multiplication,

$$\begin{pmatrix} w_1 \\ \\ \vdots \\ \\ w_N \end{pmatrix} = h^{-1} \begin{pmatrix} 0 & \frac{1}{2} & & & -\frac{1}{2} \\ -\frac{1}{2} & 0 & \ddots & & \\ & & \ddots & & \\ & & \ddots & 0 & \frac{1}{2} \\ \frac{1}{2} & & & -\frac{1}{2} & 0 \end{pmatrix} \begin{pmatrix} u_1 \\ \\ \vdots \\ \\ u_N \end{pmatrix}. \qquad (1.2)$$

(Omitted entries here and in other sparse matrices in this book are zero.) Observe that this matrix is *Toeplitz*, having constant entries along diagonals; i.e., $a_{ij}$ depends only on $i-j$. It is also *circulant*, meaning that $a_{ij}$ depends only on $(i-j) \pmod N$. The diagonals "wrap around" the matrix.

An alternative way to derive (1.1) and (1.2) is by the following process of local interpolation and differentiation:

*For* $j = 1, 2, \ldots, N$*:*

- *Let* $p_j$ *be the unique polynomial of degree* $\leq 2$ *with* $p_j(x_{j-1}) = u_{j-1}$*,* $p_j(x_j) = u_j$*, and* $p_j(x_{j+1}) = u_{j+1}$*.*
- *Set* $w_j = p_j'(x_j)$*.*

It is easily seen that, for fixed $j$, the interpolant $p_j$ is given by

$$p_j(x) = u_{j-1}a_{-1}(x) + u_j a_0(x) + u_{j+1}a_1(x),$$

where $a_{-1}(x) = (x - x_j)(x - x_{j+1})/2h^2$, $a_0(x) = -(x - x_{j-1})(x - x_{j+1})/h^2$, and $a_1(x) = (x - x_{j-1})(x - x_j)/2h^2$. Differentiating and evaluating at $x = x_j$ then gives (1.1).

This derivation by local interpolation makes it clear how we can generalize to higher orders. Here is the fourth-order analogue:

*For* $j = 1, 2, \ldots, N$*:*

- *Let* $p_j$ *be the unique polynomial of degree* $\leq 4$ *with* $p_j(x_{j\pm 2}) = u_{j\pm 2}$*,* $p_j(x_{j\pm 1}) = u_{j\pm 1}$*, and* $p_j(x_j) = u_j$*.*
- *Set* $w_j = p_j'(x_j)$*.*

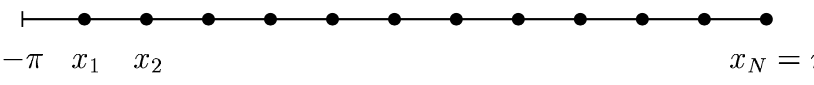
Again assuming periodicity of the data, it can be shown that this prescription amounts to the matrix-vector product

$$\begin{pmatrix} w_1 \\ \\ \\ \vdots \\ \\ \\ w_N \end{pmatrix} = h^{-1} \begin{pmatrix} & & \ddots & & & \frac{1}{12} & -\frac{2}{3} \\ & & \ddots & -\frac{1}{12} & & & \frac{1}{12} \\ & & \ddots & \frac{2}{3} & \ddots & & \\ & & \ddots & 0 & \ddots & & \\ & & \ddots & -\frac{2}{3} & \ddots & & \\ -\frac{1}{12} & & & \frac{1}{12} & \ddots & & \\ \frac{2}{3} & -\frac{1}{12} & & & \ddots & & \end{pmatrix} \begin{pmatrix} u_1 \\ \\ \\ \vdots \\ \\ \\ u_N \end{pmatrix}. \qquad (1.3)$$

This time we have a pentadiagonal instead of tridiagonal circulant matrix.

The matrices of (1.2) and (1.3) are examples of *differentiation matrices.* They have order of accuracy 2 and 4, respectively. That is, for data $u_j$ obtained by sampling a sufficiently smooth function $u$, the corresponding discrete approximations to $u'(x_j)$ will converge at the rates $O(h^2)$ and $O(h^4)$ as $h \to 0$, respectively. One can verify this by considering Taylor series.

Our first MATLAB program, Program 1, illustrates the behavior of (1.3). We take $u(x) = e^{\sin(x)}$ to give periodic data on the domain $[-\pi, \pi]$:



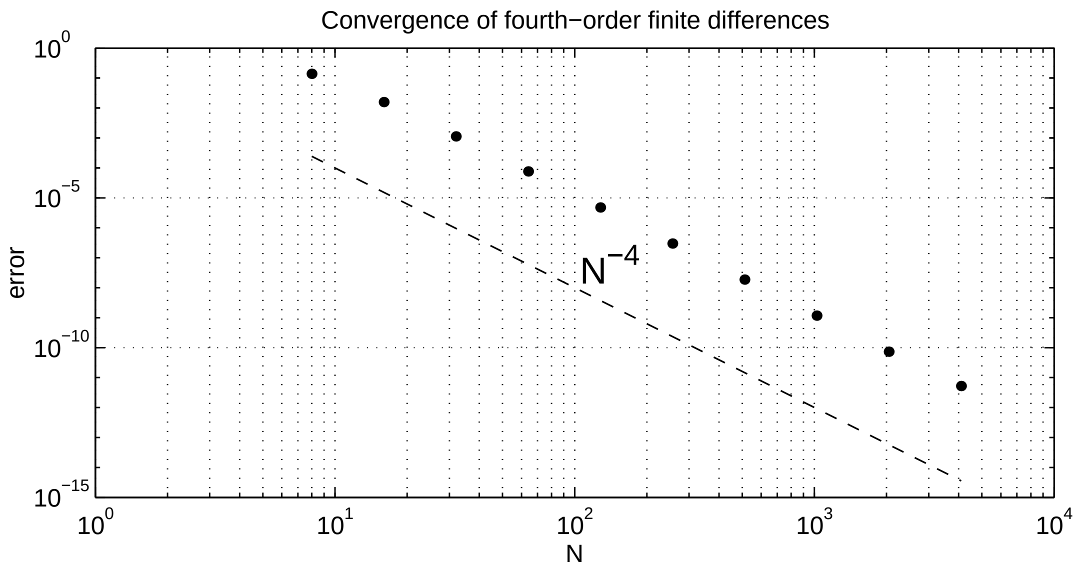
The program compares the finite difference approximation $w_j$ with the exact derivative, $e^{\sin(x_j)}\cos(x_j)$, for various values of $N$. Because it makes use of MATLAB sparse matrices, this code runs in a fraction of a second on a workstation, even though it manipulates matrices of dimensions as large as 4096 [GMS92]. The results are presented in Output 1, which plots the maximum error on the grid against $N$. The fourth-order accuracy is apparent. This is our first and last example that does not illustrate a spectral method!

We have looked at second- and fourth-order finite differences, and it is clear that consideration of sixth-, eighth-, and higher order schemes will lead to circulant matrices of increasing bandwidth. The idea behind spectral methods is to take this process to the limit, at least in principle, and work with a differentiation formula of infinite order and infinite bandwidth—i.e., a dense matrix [For75]. In the next chapter we shall show that in this limit, for an

## Program 1

```
% p1.m - convergence of fourth-order finite differences

% For various N, set up grid in [-pi,pi] and function u(x):
  Nvec = 2.^(3:12);
  clf, subplot('position',[.1 .4 .8 .5])
  for N = Nvec
    h = 2*pi/N; x = -pi + (1:N)'*h;
    u = exp(sin(x)); uprime = cos(x).*u;

    % Construct sparse fourth-order differentiation matrix:
    e = ones(N,1);
    D =   sparse(1:N,[2:N 1],2*e/3,N,N)...
        - sparse(1:N,[3:N 1 2],e/12,N,N);
    D = (D-D')/h;

    % Plot max(abs(D*u-uprime)):
    error = norm(D*u-uprime,inf);
    loglog(N,error,'.','markersize',15), hold on
  end
  grid on, xlabel N, ylabel error
  title('Convergence of fourth-order finite differences')
  semilogy(Nvec,Nvec.^(-4),'--')
  text(105,5e-8,'N^{-4}','fontsize',18)
```

## Output 1



Output 1: *Fourth-order convergence of the finite difference differentiation process* (1.3). *The use of sparse matrices permits high values of $N$.*

infinite equispaced grid, one obtains the following infinite matrix:

$$D = h^{-1}\begin{pmatrix} & \vdots & \\ \ddots & \frac{1}{3} & \\ \ddots & -\frac{1}{2} & \\ \ddots & 1 & \\ & 0 & \\ & -1 & \ddots \\ & \frac{1}{2} & \ddots \\ & -\frac{1}{3} & \ddots \\ & \vdots & \end{pmatrix}. \qquad (1.4)$$

This is a skew-symmetric ($D^T = -D$) doubly infinite Toeplitz matrix, also known as a *Laurent operator* [Hal74, Wid65]. All its entries are nonzero except those on the main diagonal.

Of course, in practice one does not work with an infinite matrix. For a finite grid, here is the design principle for spectral collocation methods:

- *Let $p$ be a single function (independent of $j$) such that $p(x_j) = u_j$ for all $j$.*
- *Set $w_j = p'(x_j)$.*

We are free to choose $p$ to fit the problem at hand. For a periodic domain, the natural choice is a trigonometric polynomial on an equispaced grid, and the resulting "Fourier" methods will be our concern through Chapter 4 and intermittently in later chapters. For nonperiodic domains, algebraic polynomials on irregular grids are the right choice, and we will describe the "Chebyshev" methods of this type beginning in Chapters 5 and 6.

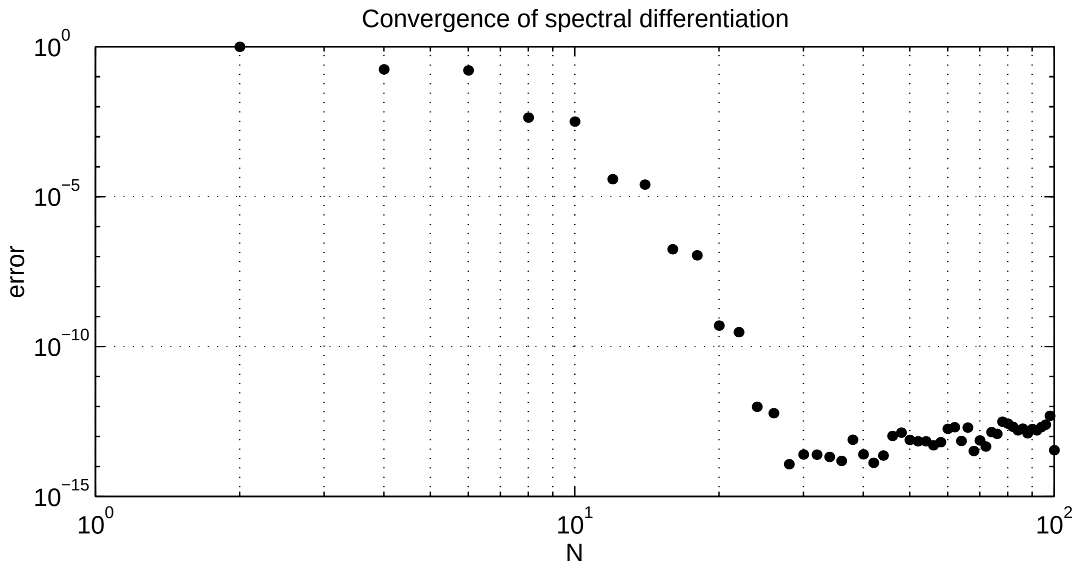
For finite $N$, taking $N$ even for simplicity, here is the $N \times N$ dense matrix we will derive in Chapter 3 for a periodic, regular grid:

$$D_N = \begin{pmatrix} & \vdots & \\ \ddots & \frac{1}{2}\cot\frac{3h}{2} & \\ \ddots & -\frac{1}{2}\cot\frac{2h}{2} & \\ \ddots & \frac{1}{2}\cot\frac{1h}{2} & \\ & 0 & \\ & -\frac{1}{2}\cot\frac{1h}{2} & \ddots \\ & \frac{1}{2}\cot\frac{2h}{2} & \ddots \\ & -\frac{1}{2}\cot\frac{3h}{2} & \ddots \\ & \vdots & \end{pmatrix}. \qquad (1.5)$$

## Program 2

```
% p2.m - convergence of periodic spectral method (compare p1.m)

% For various N (even), set up grid as before:
  clf, subplot('position',[.1 .4 .8 .5])
  for N = 2:2:100;
    h = 2*pi/N;
    x = -pi + (1:N)'*h;
    u = exp(sin(x)); uprime = cos(x).*u;

    % Construct spectral differentiation matrix:
    column = [0 .5*(-1).^(1:N-1).*cot((1:N-1)*h/2)];
    D = toeplitz(column,column([1 N:-1:2]));

    % Plot max(abs(D*u-uprime)):
    error = norm(D*u-uprime,inf);
    loglog(N,error,'.','markersize',15), hold on
  end
  grid on, xlabel N, ylabel error
  title('Convergence of spectral differentiation')
```

## Output 2



Output 2: *"Spectral accuracy" of the spectral method* (1.5)*, until the rounding errors take over around* $10^{-14}$*. Now the matrices are dense, but the values of* $N$ *are much smaller than in Program* 1*.*

A little manipulation of the cotangent function reveals that this matrix is indeed circulant as well as Toeplitz (Exercise 1.2).

Program 2 is the same as Program 1 except with (1.3) replaced by (1.5). What a difference it makes in the results! The errors in Output 2 decrease very rapidly until such high precision is achieved that rounding errors on the computer prevent any further improvement.* This remarkable behavior is called *spectral accuracy*. We will give this phrase some precision in Chapter 4, but for the moment, the point to note is how different it is from convergence rates for finite difference and finite element methods. As $N$ increases, the error in a finite difference or finite element scheme typically decreases like $O(N^{-m})$ for some constant $m$ that depends on the order of approximation and the smoothness of the solution. For a spectral method, convergence at the rate $O(N^{-m})$ for *every* $m$ is achieved, provided the solution is infinitely differentiable, and even faster convergence at a rate $O(c^N)$ $(0 < c < 1)$ is achieved if the solution is suitably analytic.

The matrices we have described have been circulant. The action of a circulant matrix is a convolution, and as we shall see in Chapter 3, convolutions can be computed using a discrete Fourier transform (DFT). Historically, it was the discovery of the fast Fourier transform (FFT) for such problems in 1965 that led to the surge of interest in spectral methods in the 1970s. We shall see in Chapter 8 that the FFT is applicable not only to trigonometric polynomials on equispaced grids, but also to algebraic polynomials on Chebyshev grids. Yet spectral methods implemented without the FFT are powerful, too, and in many applications it is quite satisfactory to work with explicit matrices. Most problems in this book are solved via matrices.

Summary of This Chapter. The fundamental principle of spectral collocation methods is, given discrete data on a grid, to interpolate the data globally, then evaluate the derivative of the interpolant on the grid. For periodic problems, we normally use trigonometric interpolants in equispaced points, and for nonperiodic problems, we normally use polynomial interpolants in unevenly spaced points.

## Exercises

**1.1.** We derived the entries of the tridiagonal circulant matrix (1.2) by local polynomial interpolation. Derive the entries of the pentadiagonal circulant matrix (1.3) in the same manner.

*All our calculations are done in standard IEEE double precision arithmetic with $\epsilon_{\text{machine}} = 2^{-53} \approx 1.11 \times 10^{-16}$. This means that each addition, multiplication, division, and subtraction produces the exactly correct result times some factor $1 + \delta$ with $|\delta| \le \epsilon_{\text{machine}}$. See [Hig96] and [TrBa97].

**1.2.** Show that (1.5) is circulant.

**1.3.** The dots of Output 2 lie in pairs. Why? What property of $e^{\sin(x)}$ gives rise to this behavior?

**1.4.** Run Program 1 to $N = 2^{16}$ instead of $2^{12}$. What happens to the plot of the error vs. $N$? Why? Use the MATLAB commands `tic` and `toc` to generate a plot of approximately how the computation time depends on $N$. Is the dependence linear, quadratic, or cubic?

**1.5.** Run Programs 1 and 2 with $e^{\sin(x)}$ replaced by (a) $e^{\sin^2(x)}$ and (b) $e^{\sin(x)|\sin(x)|}$ and with `uprime` adjusted appropriately. What rates of convergence do you observe? Comment.

**1.6.** By manipulating Taylor series, determine the constant $C$ for an error expansion of (1.3) of the form $w_j - u'(x_j) \sim Ch^4 u^{(5)}(x_j)$, where $u^{(5)}$ denotes the fifth derivative. Based on this value of $C$ and on the formula for $u^{(5)}(x)$ with $u(x) = e^{\sin(x)}$, determine the leading term in the expansion for $w_j - u'(x_j)$ for $u(x) = e^{\sin(x)}$. (You will have to find $\max_{x\in[-\pi,\pi]} |u^{(5)}(x)|$ numerically.) Modify Program 1 so that it plots the dashed line corresponding to this leading term rather than just $N^{-4}$. This adjusted dashed line should fit the data almost perfectly. Plot the difference between the two on a log-log scale and verify that it shrinks at the rate $O(h^6)$.

# 6. Chebyshev Differentiation Matrices

In the last chapter we discussed why grid points must cluster at boundaries for spectral methods based on polynomials. In particular, we introduced the Chebyshev points,

$$x_j = \cos(j\pi/N), \qquad j = 0, 1, \ldots, N, \tag{6.1}$$

which cluster as required. In this chapter we shall use these points to construct Chebyshev differentiation matrices and apply these matrices to differentiate a few functions. The same set of points will continue to be the basis of many of our computations throughout the rest of the book.

Our scheme is as follows. Given a grid function $v$ defined on the Chebyshev points, we obtain a discrete derivative $w$ in two steps:

- *Let $p$ be the unique polynomial of degree $\le N$ with $p(x_j) = v_j$, $0 \le j \le N$.*
- *Set $w_j = p'(x_j)$.*

This operation is linear, so it can be represented by multiplication by an $(N+1) \times (N+1)$ matrix, which we shall denote by $D_N$:

$$w = D_N v.$$

Here $N$ is an arbitrary positive integer, even or odd. The restriction to even $N$ in this book (p. 18) applies to Fourier, not Chebyshev spectral methods.

To get a feel for the interpolation process, we take a look at $N = 1$ and $N = 2$ before proceeding to the general case.

Consider first $N = 1$. The interpolation points are $x_0 = 1$ and $x_1 = -1$, and the interpolating polynomial through data $v_0$ and $v_1$, written in Lagrange form, is

$$p(x) = \tfrac{1}{2}(1+x)v_0 + \tfrac{1}{2}(1-x)v_1.$$

Taking the derivative gives

$$p'(x) = \tfrac{1}{2}v_0 - \tfrac{1}{2}v_1.$$

This formula implies that $D_1$ is the $2\times 2$ matrix whose first column contains constant entries $1/2$ and whose second column contains constant entries $-1/2$:

$$D_1 = \begin{pmatrix} \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}.$$

Now consider $N = 2$. The interpolation points are $x_0 = 1$, $x_1 = 0$, and $x_2 = -1$, and the interpolant is the quadratic

$$p(x) \;=\; \tfrac{1}{2}x(1+x)v_0 + (1+x)(1-x)v_1 + \tfrac{1}{2}x(x-1)v_2.$$

The derivative is now a linear polynomial,

$$p'(x) = (x+\tfrac{1}{2})v_0 - 2xv_1 + (x-\tfrac{1}{2})v_2.$$

The differentiation matrix $D_2$ is the $3\times 3$ matrix whose $j$th column is obtained by sampling the $j$th term of this expression at $x = 1$, $0$, and $-1$:

$$D_2 = \begin{pmatrix} \frac{3}{2} & -2 & \frac{1}{2} \\ \frac{1}{2} & 0 & -\frac{1}{2} \\ -\frac{1}{2} & 2 & -\frac{3}{2} \end{pmatrix}. \tag{6.2}$$

It is no coincidence that the middle row of this matrix contains the coefficients for a centered three-point finite difference approximation to a derivative, and the other rows contain the coefficients for one-sided approximations such as the one that drives the second-order Adams–Bashforth formula for the numerical solution of ODEs [For88]. The rows of higher order spectral differentiation matrices can also be viewed as vectors of coefficients of finite difference formulas, but these will be based on uneven grids and thus no longer familiar from standard applications.

We now give formulas for the entries of $D_N$ for arbitrary $N$. These were first published perhaps in [GHO84] and are derived in Exercises 6.1 and 6.2. Analogous formulas for general sets $\{x_j\}$ rather than just Chebyshev points are stated in Exercise 6.1.

Theorem 7 Chebyshev differentiation matrix.

*For each $N \geq 1$, let the rows and columns of the $(N+1) \times (N+1)$ Chebyshev spectral differentiation matrix $D_N$ be indexed from $0$ to $N$. The entries of this matrix are*

$$(D_N)_{00} = \frac{2N^2+1}{6}, \qquad (D_N)_{NN} = -\frac{2N^2+1}{6}, \tag{6.3}$$

$$(D_N)_{jj} = \frac{-x_j}{2(1-x_j^2)}, \qquad j = 1, \ldots, N-1, \tag{6.4}$$

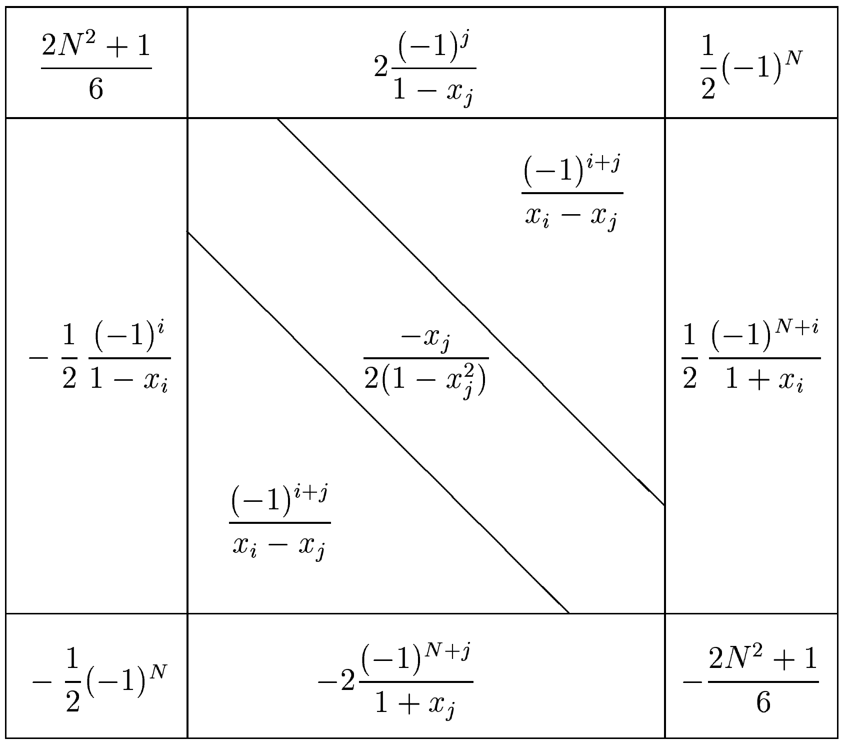
$$(D_N)_{ij} = \frac{c_i}{c_j} \frac{(-1)^{i+j}}{(x_i - x_j)}, \qquad i \neq j, \quad i,j = 0, \ldots, N, \tag{6.5}$$

*where*

$$c_i = \begin{cases} 2, & i = 0 \textit{ or } N, \\ 1, & \textit{otherwise.} \end{cases}$$

A picture makes the pattern clearer:

$D_N \;=$


$\frac{2N^2+1}{6}$
$2\frac{(-1)^j}{1-x_j}$
$\frac{1}{2}(-1)^N$
$\frac{(-1)^{i+j}}{x_i - x_j}$
$-\frac{1}{2}\frac{(-1)^i}{1-x_i}$
$\frac{-x_j}{2(1-x_j^2)}$
$\frac{1}{2}\frac{(-1)^{N+i}}{1+x_i}$
$\frac{(-1)^{i+j}}{x_i - x_j}$
$-\frac{1}{2}(-1)^N$
$-2\frac{(-1)^{N+j}}{1+x_j}$
$-\frac{2N^2+1}{6}$

The $j$th column of $D_N$ contains the derivative of the degree $N$ polynomial interpolant $p_j(x)$ to the delta function supported at $x_j$, sampled at the grid

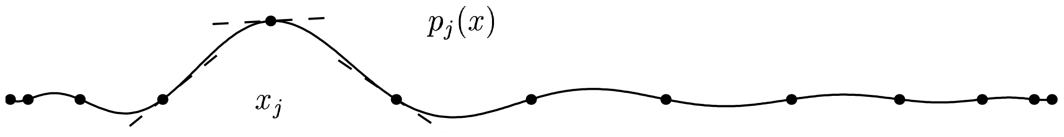


Fig. 6.1. *Degree* 12 *polynomial interpolant* $p(x)$ *to the delta function supported at* $x_8$ *on the* 13*-point Chebyshev grid with* $N = 12$. *The slopes indicated by the dashed lines, from right to left, are the entries* $(D_{12})_{7,8}$, $(D_{12})_{8,8}$, *and* $(D_{12})_{9,8}$ *of the* $13 \times 13$ *spectral differentiation matrix* $D_{12}$.

points $\{x_i\}$. Three such sampled values are suggested by the dashed lines in Figure 6.1.

Throughout this text, we take advantage of MATLAB's high-level commands for such operations as polynomial interpolation, matrix inversion, and FFT. For clarity of exposition, as explained in the "Note on the MATLAB Programs" at the beginning of the book, our style is to make our programs short and self-contained. However, there will be one major exception to this rule, one MATLAB function that we will define and then call repeatedly whenever we need Chebyshev grids and differentiation matrices. The function is called `cheb`, and it returns a vector $x$ and a matrix $D$.

## cheb.m

```
% CHEB  compute D = differentiation matrix, x = Chebyshev grid

  function [D,x] = cheb(N)
  if N==0, D=0; x=1; return, end
  x = cos(pi*(0:N)/N)';
  c = [2; ones(N-1,1); 2].*(-1).^(0:N)';
  X = repmat(x,1,N+1);
  dX = X-X';
  D  = (c*(1./c)')./(dX+(eye(N+1)));      % off-diagonal entries
  D  = D - diag(sum(D'));                 % diagonal entries
```

Note that this program does not compute $D_N$ exactly by formulas (6.3)–(6.5). It utilizes (6.5) for the off-diagonal entries but then obtains the diagonal entries (6.3)–(6.4) from the identity

$$(D_N)_{ii} = -\sum_{\substack{j=0 \\ j\neq i}}^{N} (D_N)_{ij}. \tag{6.6}$$

This is marginally simpler to program, and it produces a matrix with better stability properties in the presence of rounding errors [BaBe00, BCM94]. Equation (6.6) can be derived by noting that the interpolant to $(1, 1, \ldots, 1)^T$ is the constant function $p(x) = 1$, and since $p'(x) = 0$ for all $x$, $D_N$ must map $(1, 1, \ldots, 1)^T$ to the zero vector.

Here are the first five Chebyshev differentiation matrices as computed by `cheb`. Note that they are dense, with little apparent structure apart from the antisymmetry condition $(D_N)_{ij} = -(D_N)_{N-i,N-j}$.

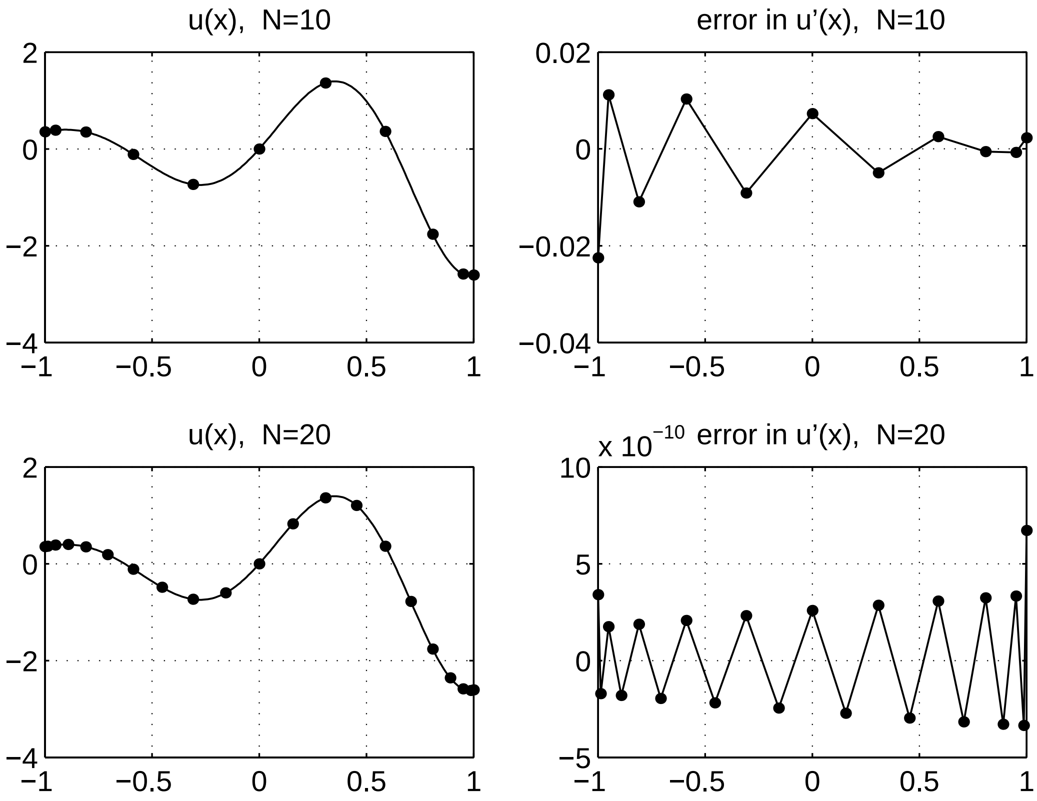
```
>> cheb(1)
    0.5000   -0.5000
    0.5000   -0.5000

>> cheb(2)
    1.5000   -2.0000    0.5000
    0.5000   -0.0000   -0.5000
   -0.5000    2.0000   -1.5000

>> cheb(3)
    3.1667   -4.0000    1.3333   -0.5000
    1.0000   -0.3333   -1.0000    0.3333
   -0.3333    1.0000    0.3333   -1.0000
    0.5000   -1.3333    4.0000   -3.1667

>> cheb(4)
    5.5000   -6.8284    2.0000   -1.1716    0.5000
    1.7071   -0.7071   -1.4142    0.7071   -0.2929
   -0.5000    1.4142   -0.0000   -1.4142    0.5000
    0.2929   -0.7071    1.4142    0.7071   -1.7071
   -0.5000    1.1716   -2.0000    6.8284   -5.5000

>> cheb(5)
    8.5000  -10.4721    2.8944   -1.5279    1.1056   -0.5000
    2.6180   -1.1708   -2.0000    0.8944   -0.6180    0.2764
   -0.7236    2.0000   -0.1708   -1.6180    0.8944   -0.3820
    0.3820   -0.8944    1.6180    0.1708   -2.0000    0.7236
   -0.2764    0.6180   -0.8944    2.0000    1.1708   -2.6180
    0.5000   -1.1056    1.5279   -2.8944   10.4721   -8.5000
```

Program 11 illustrates how $D_N$ can be used to differentiate the smooth, nonperiodic function $u(x) = e^x \sin(5x)$ on grids with $N = 10$ and $N = 20$. The output shows a graph of $u(x)$ alongside a plot of the error in $u'(x)$. With $N = 20$, we get nine-digit accuracy.

## Program 11

```
% p11.m - Chebyshev differentation of a smooth function

  xx = -1:.01:1; uu = exp(xx).*sin(5*xx); clf
  for N = [10 20]
    [D,x] = cheb(N); u = exp(x).*sin(5*x);
      subplot('position',[.15 .66-.4*(N==20) .31 .28])
      plot(x,u,'.','markersize',14), grid on
      line(xx,uu)
      title(['u(x),  N=' int2str(N)])
    error = D*u - exp(x).*(sin(5*x)+5*cos(5*x));
      subplot('position',[.55 .66-.4*(N==20) .31 .28])
      plot(x,error,'.','markersize',14), grid on
      line(x,error)
      title(['  error in u''(x),  N=' int2str(N)])
  end
```

## Output 11



Output 11: *Chebyshev differentiation of* $u(x) = e^x \sin(5x)$. *Note the vertical scales.*

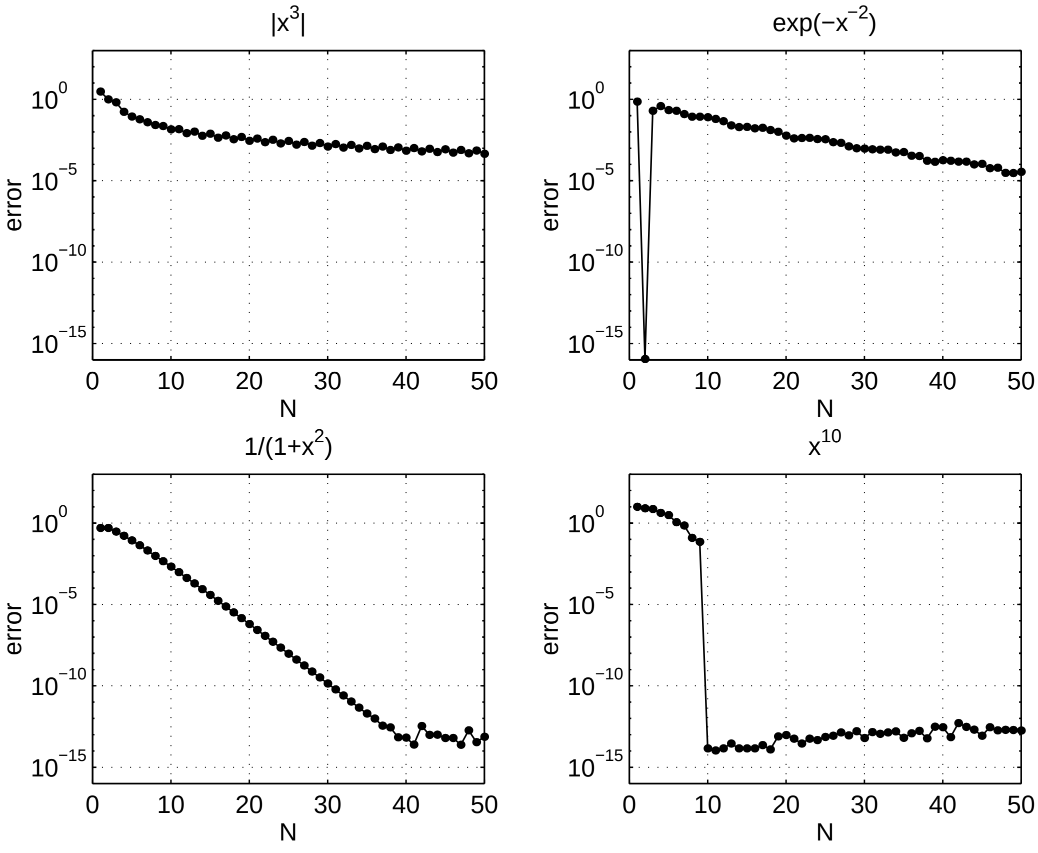
## Program 12

```
% p12.m - accuracy of Chebyshev spectral differentiation
%         (compare p7.m)

% Compute derivatives for various values of N:
  Nmax = 50; E = zeros(3,Nmax);
  for N = 1:Nmax;
    [D,x] = cheb(N);
    v = abs(x).^3; vprime = 3*x.*abs(x);     % 3rd deriv in BV
    E(1,N) = norm(D*v-vprime,inf);
    v = exp(-x.^(-2)); vprime = 2.*v./x.^3;  % C-infinity
    E(2,N) = norm(D*v-vprime,inf);
    v = 1./(1+x.^2); vprime = -2*x.*v.^2;    % analytic in [-1,1]
    E(3,N) = norm(D*v-vprime,inf);
    v = x.^10; vprime = 10*x.^9;             % polynomial
    E(4,N) = norm(D*v-vprime,inf);
  end

% Plot results:
  titles = {'|x^3|','exp(-x^{-2})','1/(1+x^2)','x^{10}'}; clf
  for iplot = 1:4
    subplot(2,2,iplot)
    semilogy(1:Nmax,E(iplot,:),'.','markersize',12)
    line(1:Nmax,E(iplot,:))
    axis([0 Nmax 1e-16 1e3]), grid on
    set(gca,'xtick',0:10:Nmax,'ytick',(10).^(-15:5:0))
    xlabel N, ylabel error, title(titles(iplot))
  end
```

Program 12, the Chebyshev analogue of Program 7, illustrates spectral accuracy more systematically. Four functions are spectrally differentiated: $|x^3|$, $\exp(-x^{-2})$, $1/(1+x^2)$, and $x^{10}$. The first has a third derivative of bounded variation, the second is smooth but not analytic, the third is analytic in a neighborhood of $[-1,1]$, and the fourth is a polynomial, the analogue for Chebyshev spectral methods of a band-limited function for Fourier spectral methods.

Summary of This Chapter. The entries of the Chebyshev differentiation matrix $D_N$ can be computed by explicit formulas, which can be conveniently collected in an eight-line MATLAB function. More general explicit formulas can be used to construct the differentiation matrix for an arbitrarily prescribed set of distinct points $\{x_j\}$.

## Output 12



Output 12: *Accuracy of the Chebyshev spectral derivative for four functions of increasing smoothness. Compare Output* 7 *(p.* 36*).*

## Exercises

**6.1.** If $x_0, x_1, \ldots, x_N \in \mathbb{R}$ are distinct, then the *cardinal function* $p_j(x)$ defined by

$$p_j(x) = \frac{1}{a_j}\prod_{\substack{k=0\\k\neq j}}^{N}(x-x_k), \qquad a_j = \prod_{\substack{k=0\\k\neq j}}^{N}(x_j-x_k) \tag{6.7}$$

is the unique polynomial interpolant of degree $N$ to the values 1 at $x_j$ and 0 at $x_k$, $k \neq j$. Take the logarithm and differentiate to obtain

$$p_j'(x) = p_j(x)\sum_{\substack{k=0\\k\neq j}}^{N}(x-x_k)^{-1},$$

and from this derive the formulas

$$D_{ij} = \frac{1}{a_j}\prod_{\substack{k=0\\k\neq i,j}}^{N}(x_i - x_k) = \frac{a_i}{a_j(x_i - x_j)} \qquad (i \neq j) \tag{6.8}$$

and

$$D_{jj} = \sum_{\substack{k=0\\k\neq j}}^{N}(x_j - x_k)^{-1} \tag{6.9}$$

for the entries of the $N \times N$ differentiation matrix associated with the points $\{x_j\}$. (See also Exercise 12.2.)

**6.2.** Derive Theorem 7 from (6.8) and (6.9).

**6.3.** Suppose $1 = x_0 > x_1 > \cdots > x_N = -1$ lie in the minimal-energy configuration in $[-1, 1]$ in the sense discussed on p. 49. Show that except in the corners, the diagonal entries of the corresponding differentiation matrix $D$ are zero.

**6.4.** It was mentioned on p. 55 that Chebyshev differentiation matrices have the symmetry property $(D_N)_{ij} = -(D_N)_{N-i,N-j}$. (a) Explain where this condition comes from. (b) Derive the analogous symmetry condition for $(D_N)^2$. (c) Taking $N$ to be odd, so that the dimension of $D_N$ is even, explain how $(D_N)^2$ could be constructed from just half the entries of $D_N$. For large $N$, how does the floating point operation count for this process compare with that for straightforward squaring of $D_N$?

**6.5.** Modify `cheb` so that it computes the diagonal entries of $D_N$ by the explicit formulas (6.3)–(6.4) rather than by (6.6). Confirm that your code produces the same results except for rounding errors. Then see if you can find numerical evidence that it is less stable numerically than `cheb`.

**6.6.** The second panel of Output 12 shows a sudden dip for $N = 2$. Show that in fact, $E(2, 2) = 0$ (apart from rounding errors).

**6.7.** Theorem 6 makes a prediction about the geometric rate of convergence in the third panel of Output 12. Exactly what is this prediction? How well does it match the observed rate of convergence?

**6.8.** Let $D_N$ be the usual Chebyshev differentiation matrix. Show that the power $(D_N)^{N+1}$ is identically equal to zero. Now try it on the computer for $N = 5$ and 20 and report the computed 2-norms $\|(D_5)^6\|_2$ and $\|(D_{20})^{21}\|_2$. Discuss.

# 9. Eigenvalues and Pseudospectra

Spectral methods are powerful tools for the computation of eigenvalues of differential and integral operators and their generalizations for strongly nonsymmetric problems, pseudospectra. Indeed, it was Orszag's 1971 computation of the critical Reynolds number $R = 5772.22$ for eigenvalue instability of plane Poiseuille fluid flow, a problem we shall discuss in Chapter 14, that did as much as anything to establish spectral methods as an important tool in scientific computing [Ors71]. Perhaps the reason why spectral methods are so important for eigenvalue computations is that these are applications where high accuracy tends to be crucial.

So far in this book we have seen two examples of eigenvalue calculations. Program 8 (p. 38) solved the harmonic oscillator problem

$$-u_{xx} + x^2 u \;=\; \lambda u, \qquad x \in \mathbb{R},$$

by a Fourier spectral method, taking advantage of the exponential decay of the eigenfunctions to replace the real line $\mathbb{R}$ by the periodic interval $[-L, L]$. Program 15 (p. 66) solved the even simpler problem

$$u_{xx} = \lambda u, \qquad -1 < x < 1, \;\; u(\pm 1) = 0$$

by a Chebyshev spectral method that imposed the homogeneous Dirichlet conditions explicitly. In this chapter, we will develop such methods further with the aid of four additional examples. In each case we apply spectral ideas via matrices rather than the FFT, since it is so convenient to take advantage of the standard powerful algorithms for matrix eigenvalue and generalized eigenvalue problems embodied in the MATLAB commands `eig` and `eigs`.

Our four examples and the special features they illustrate can be summarized as follows:

| | | |
|---|---|---|
| Program 21: | Mathieu equation, | periodic domain; |
| Program 22: | Airy equation, | generalized eigenvalue problem; |
| Program 23: | membrane oscillations, | two-dimensional domain; |
| Program 24: | complex harmonic oscillator, | pseudospectra. |

We begin with the Mathieu equation, an ODE that arises in problems of forced oscillations. (This example has been taken from Weideman and Reddy [WeRe00].) The equation can be written

$$-u_{xx} + 2q\cos(2x)u \;=\; \lambda u, \tag{9.1}$$

where $q$ is a real parameter, and we look for periodic solutions on $[-\pi,\pi]$. For $q=0$, we have the linear pendulum equation of Program 15, with eigenvalues $n^2/4$ for $n=1,2,3,\dots$. The scientific interest arises in the behavior of these eigenvalues as $q$ is increased.

To compute eigenvalues of the Mathieu equation by a spectral method, Program 21 discretizes (9.1) in a routine fashion. Translating the equation from the domain $[-\pi,\pi]$ to $[0,2\pi]$ leaves the eigenvalues unaltered, so our discretization takes the form

$$L_N = -D_N^{(2)} + 2q\,\mathrm{diag}(\cos(2x_1),\dots,\cos(2x_N)),$$

where $D_N^{(2)}$ is the second-order Fourier differentiation matrix. The computation is straightforward, and with $N=42$, we get about 13 digits of accuracy. Output 21 presents the plot generated by this program, showing the curves traced by the first 11 eigenvalues as $q$ increases from 0 to 15. Producing this image took about half a second on my workstation. As shown in the figure, it is almost identical to Figure 20.1 on p. 724 of the classic *Handbook of Mathematical Functions* published by Abramowitz and Stegun in the 1960s [AbSt65]. (A few imperfections in the *Handbook* plot can be discerned, for example in the slope of the $a_2$ curve at $q=0$.) We do not know how many seconds it took Gertrude Blanch, the author of the chapter on Mathieu functions in the *Handbook*, to produce that figure.

For our second example we turn to another classical problem of applied mathematics, the Airy equation [AbSt65, BeOr78]. Traditionally, the Airy equation is posed on the real line,

$$u_{xx} = xu, \qquad x\in\mathbb{R}. \tag{9.2}$$

This is the canonical example of an ODE that changes type in different parts of the domain. For $x<0$, the behavior is oscillatory, while for $x>0$, we get growing and decaying exponential solutions. Being an ODE of second order, the Airy equation has a two-dimensional linear space of solutions, and the

## Program 21

```
% p21.m - eigenvalues of Mathieu operator -u_xx + 2qcos(2x)u
%         (compare p8.m and p. 724 of Abramowitz & Stegun)

  N = 42; h = 2*pi/N; x = h*(1:N);
  D2 = toeplitz([-pi^2/(3*h^2)-1/6 ...
                -.5*(-1).^(1:N-1)./sin(h*(1:N-1)/2).^2]);
  qq = 0:.2:15; data = [];
  for q = qq;
    e = sort(eig(-D2 + 2*q*diag(cos(2*x))))';
    data = [data; e(1:11)];
  end
  clf, subplot(1,2,1)
  set(gca,'colororder',[0 0 1],'linestyleorder','-|--'), hold on
  plot(qq,data), xlabel q, ylabel \lambda
  axis([0 15 -24 32]), set(gca,'ytick',-24:4:32)
```

## Output 21

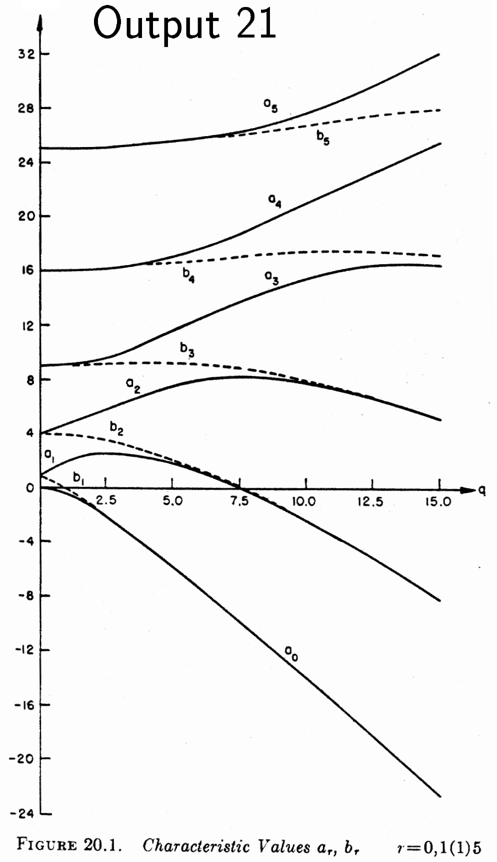
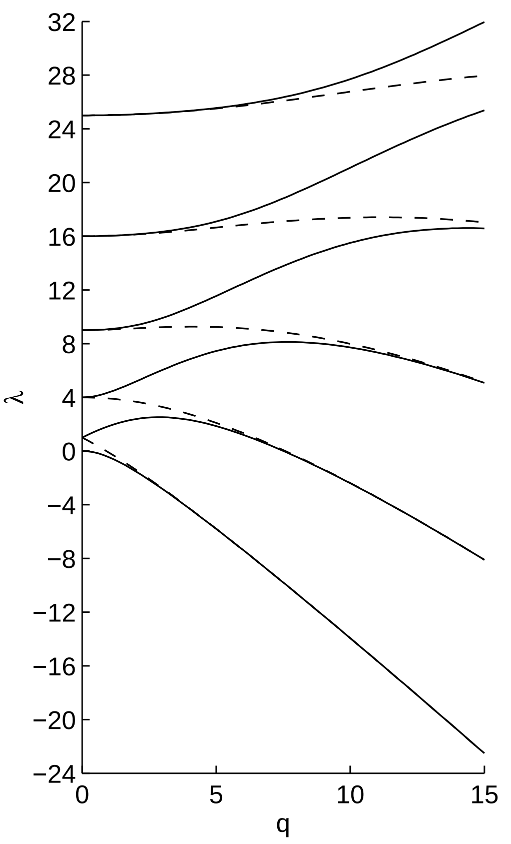


FIGURE 20.1. *Characteristic Values $a_r$, $b_r$ $r=0,1(1)5$*



Output 21: *The first* 11 *eigenvalues of the Mathieu equation* (9.1)*. Left, Figure* 20.1 *from Abramowitz and Stegun* (1965)*. Right, output from Program* 21*.*

standard basis for this space is the pair of Airy functions $\mathrm{Ai}(x)$, which decays exponentially as $x \to \infty$, and $\mathrm{Bi}(x)$, which grows exponentially.

To solve the Airy equation numerically on the real line by spectral methods is not entirely straightforward, because there are an infinite number of oscillations that decay only algebraically in amplitude; there is an essential singularity at $-\infty$. Instead, in Program 22 we consider a slightly different eigenvalue problem posed on a finite interval,

$$u_{xx} = \lambda x u, \qquad u(\pm 1) = 0, \quad -1 < x < 1. \tag{9.3}$$

This differs from our previous eigenvalue problems in a basic way: rather than being of the form (in linear algebra notation) $Au = \lambda u$, it is a *generalized eigenvalue problem* of the form $Au = \lambda Bu$. If $B$ is nonsingular, then a generalized eigenvalue problem can be reduced mathematically to a standard one, $B^{-1}Au = \lambda u$. However, this is not necessarily a good idea in practice, and if $B$ is singular, it is impossible even in principle. Instead, alternative numerical methods are generally used that deal with the generalized problem directly. The most standard is known as the QZ algorithm [GoVa96]. In MATLAB one calls upon the QZ algorithm by writing `eig(A,B)` instead of `eig(A)`. We shall not give details.

To discretize (9.3), we use a standard Chebyshev formulation for the second derivative and a diagonal matrix for the pointwise multiplication:

$$Au = \lambda Bu, \qquad A = \tilde{D}_N^2, \quad B = \mathrm{diag}(x_0, \ldots, x_N).$$

The computation is straightforward, and it is evident from Output 22 that we have spectral convergence, with ten or more digits of accuracy in the fifth eigenvector.

Figure 9.1 makes the connection back to Airy functions. In solving (9.3) we have imposed the condition $u(-1) = 0$ and looked for the fifth positive eigenvalue. This is equivalent to computing $\mathrm{Ai}(x)$ on the interval $[-L, L]$, where $-L = -7.944133\ldots$ is the location of the fifth zero of $\mathrm{Ai}(x)$. A rescaling back to $[-1, 1]$ then introduces a power $L^3 = 501.348\ldots$, and that is why our eigenvalue came out as $L^3$. Actually, what we have just said is not exactly true, for it assumed $\mathrm{Ai}(+L) = 0$, whereas in fact, $\mathrm{Ai}(L) \simeq 5.5 \times 10^{-8}$. What we have computed is actually a spectrally accurate approximation of a linear combination of $\mathrm{Ai}(Lx)$ plus a very small multiple, of order $10^{-14}$, of $\mathrm{Bi}(Lx)$. The coefficient $10^{-14}$ arises because $\mathrm{Bi}(L) \simeq 10^6$.

Our third example is a Laplace eigenvalue problem in two space dimensions,

$$-\Delta u + f(x, y)u \;=\; \lambda u, \quad -1 < x, y < 1, \quad u = 0 \text{ on the boundary.} \tag{9.4}$$

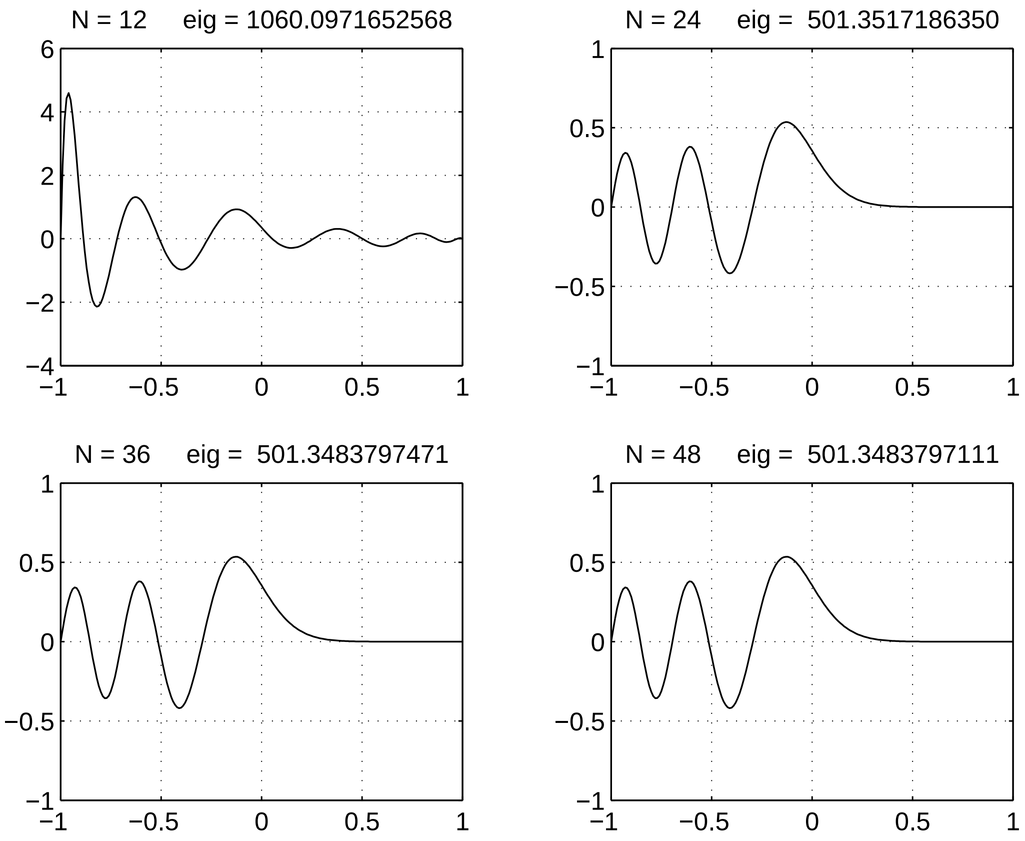
For $f = 0$, we have a familiar problem easily solvable by separation of variables: the eigenfunctions have the form

$$\sin(k_x(x+1)) \sin(k_y(y+1)),$$

## Program 22

```
% p22.m - 5th eigenvector of Airy equation u_xx = lambda*x*u

  clf
  for N = 12:12:48
    [D,x] = cheb(N); D2 = D^2; D2 = D2(2:N,2:N);
    [V,Lam] = eig(D2,diag(x(2:N)));      % generalized ev problem
    Lam = diag(Lam); ii = find(Lam>0);
    V = V(:,ii); Lam = Lam(ii);
    [foo,ii] = sort(Lam); ii = ii(5); lambda = Lam(ii);
    v = [0;V(:,ii);0]; v = v/v(N/2+1)*airy(0);
    xx = -1:.01:1; vv = polyval(polyfit(x,v,N),xx);
    subplot(2,2,N/12), plot(xx,vv), grid on
    title(sprintf('N = %d     eig = %15.10f',N,lambda))
  end
```

## Output 22



Output 22: *Convergence to the fifth eigenvector of the Airy problem* (9.3).

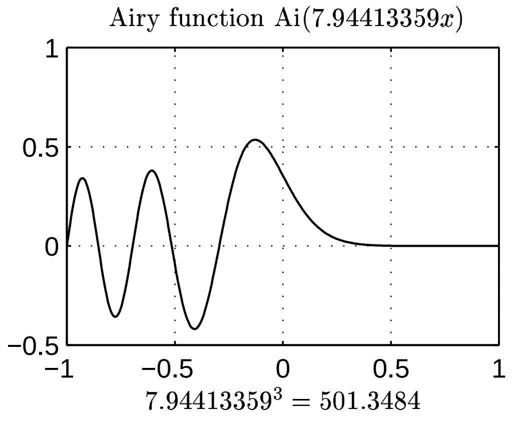


Fig. 9.1. *A rescaled solution of the Airy equation,* $\mathrm{Ai}(\lambda^{1/3}x)$. *This differs from the solution of Output* 22 *by about* $10^{-8}$.

where $k_x$ and $k_y$ are integer multiples of $\pi/2$. This gives eigenvalues

$$\frac{\pi^2}{4}(i^2+j^2), \qquad i,j=1,2,3,\ldots.$$

Note that most of the eigenvalues are degenerate: whenever $i \neq j$, the eigenvalue has multiplicity 2. For $f \neq 0$, on the other hand, (9.4) will have no analytic solution in general and the eigenvalues will not be degenerate. Perturbations will split the double eigenvalues into pairs, a phenomenon familiar to physicists.

To solve (9.4) numerically by a spectral method, we can proceed just as in Program 16 (p. 70). We again set up the discrete Laplacian (7.5) of dimension $(N-1)^2 \times (N-1)^2$ as a Kronecker sum. To this we add a diagonal matrix consisting of the perturbation $f$ evaluated at each of the $(N-1)^2$ points of the grid in the lexicographic ordering described on p. 68. The result is a large matrix whose eigenvalues can be found by standard techniques. In Program 23, this is done by MATLAB's command `eig`. For large enough problems, it would be important to use instead a Krylov subspace iterative method such as the Arnoldi or (if the matrix is symmetric) Lanczos iterations, which are implemented within MATLAB in the alternative code `eigs` (Exercise 9.4).

Output 23a shows results from Program 23 for the unperturbed case, computed by executing the code exactly as printed except with the line `L = L + diag(...)` commented out. Contour plots are given of the first four eigenmodes, with eigenvalues equal to $\pi^2/4$ times 2, 5, 5, and 8. As predicted, two of the eigenmodes are degenerate. As always in cases of degenerate eigenmodes, the choice of eigenvectors here is arbitrary. For essentially arbitrary reasons, the computation picks an eigenmode with a nodal line approximately along a diagonal; it then computes a second eigenmode linearly independent

## Program 23

```
% p23.m - eigenvalues of perturbed Laplacian on [-1,1]x[-1,1]
%         (compare p16.m)

% Set up tensor product Laplacian and compute 4 eigenmodes:
  N = 16; [D,x] = cheb(N); y = x;
  [xx,yy] = meshgrid(x(2:N),y(2:N)); xx = xx(:); yy = yy(:);
  D2 = D^2; D2 = D2(2:N,2:N); I = eye(N-1);
  L = -kron(I,D2) - kron(D2,I);                %  Laplacian
  L = L + diag(exp(20*(yy-xx-1)));             %  + perturbation
  [V,D] = eig(L); D = diag(D);
  [D,ii] = sort(D); ii = ii(1:4); V = V(:,ii);

% Reshape them to 2D grid, interpolate to finer grid, and plot:
  [xx,yy] = meshgrid(x,y);
  fine = -1:.02:1; [xxx,yyy] = meshgrid(fine,fine);
  uu = zeros(N+1,N+1);
  [ay,ax] = meshgrid([.56 .04],[.1 .5]); clf
  for i = 1:4
    uu(2:N,2:N) = reshape(V(:,i),N-1,N-1);
    uu = uu/norm(uu(:),inf);
    uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
    subplot('position',[ax(i) ay(i) .38 .38])
    contour(fine,fine,uuu,-.9:.2:.9)
    colormap([0 0 0]); axis square
    title(['eig = ' num2str(D(i)/(pi^2/4),'%18.12f') '\pi^2/4'])
  end
```

of the first (though not orthogonal to it), with a nodal line approximately on the opposite diagonal. An equally valid pair of eigenmodes in this degenerate case would have had nodal lines along the $x$ and $y$ axes.

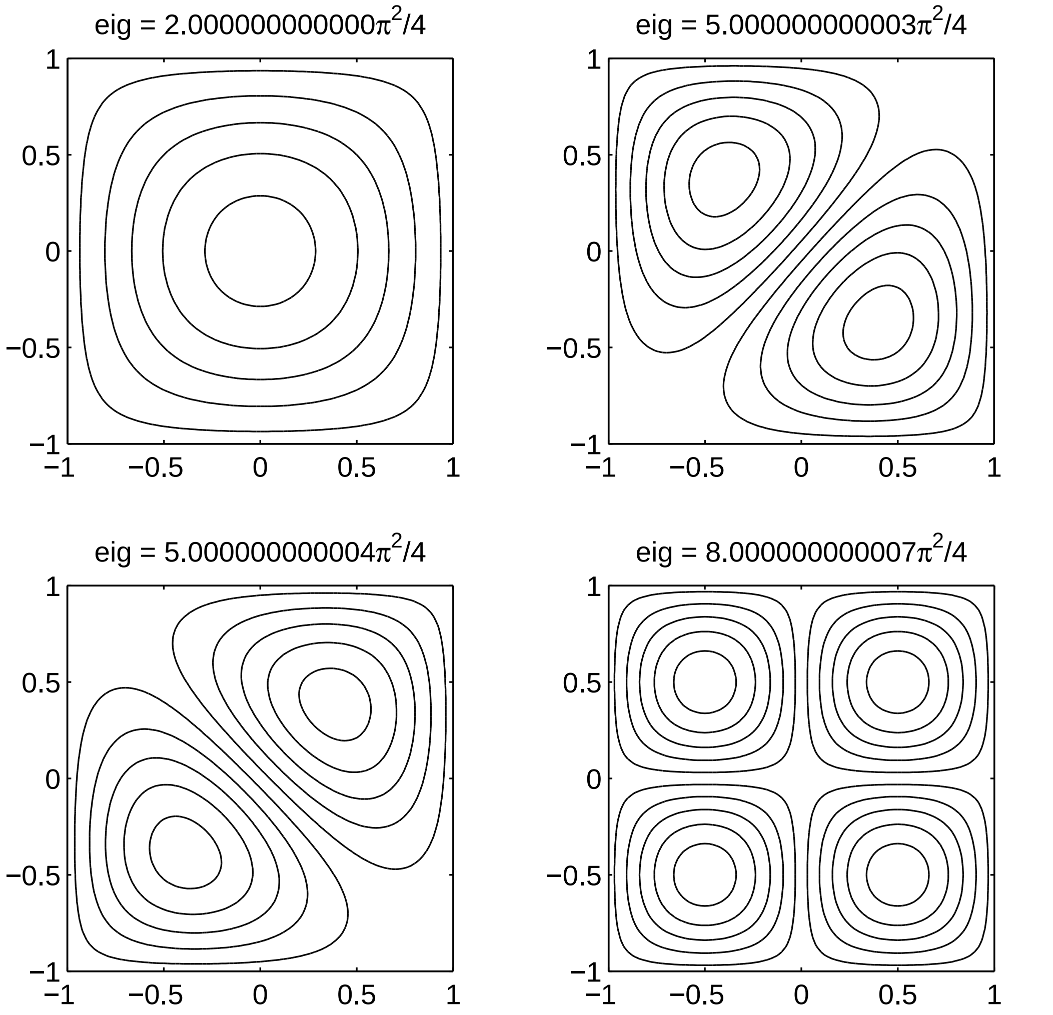
A remarkable feature of Output 23a is that although the grid is only of size $16 \times 16$, the eigenvalues are computed to 12-digit accuracy. This reflects the fact that one or two oscillations of a sine wave can be approximated to better than 12-digit precision by a polynomial of degree 16 (Exercise 9.1).

Output 23b presents the same plot with the perturbation in (9.4) included, with

$$f(x,y) = \exp(20(y-x-1)).$$

This perturbation has a very special form. It is nearly zero outside the upper left triangular region, one-eighth of the total domain, defined by $y - x \geq 1$. Within that region, however, it is very large, achieving values as great as

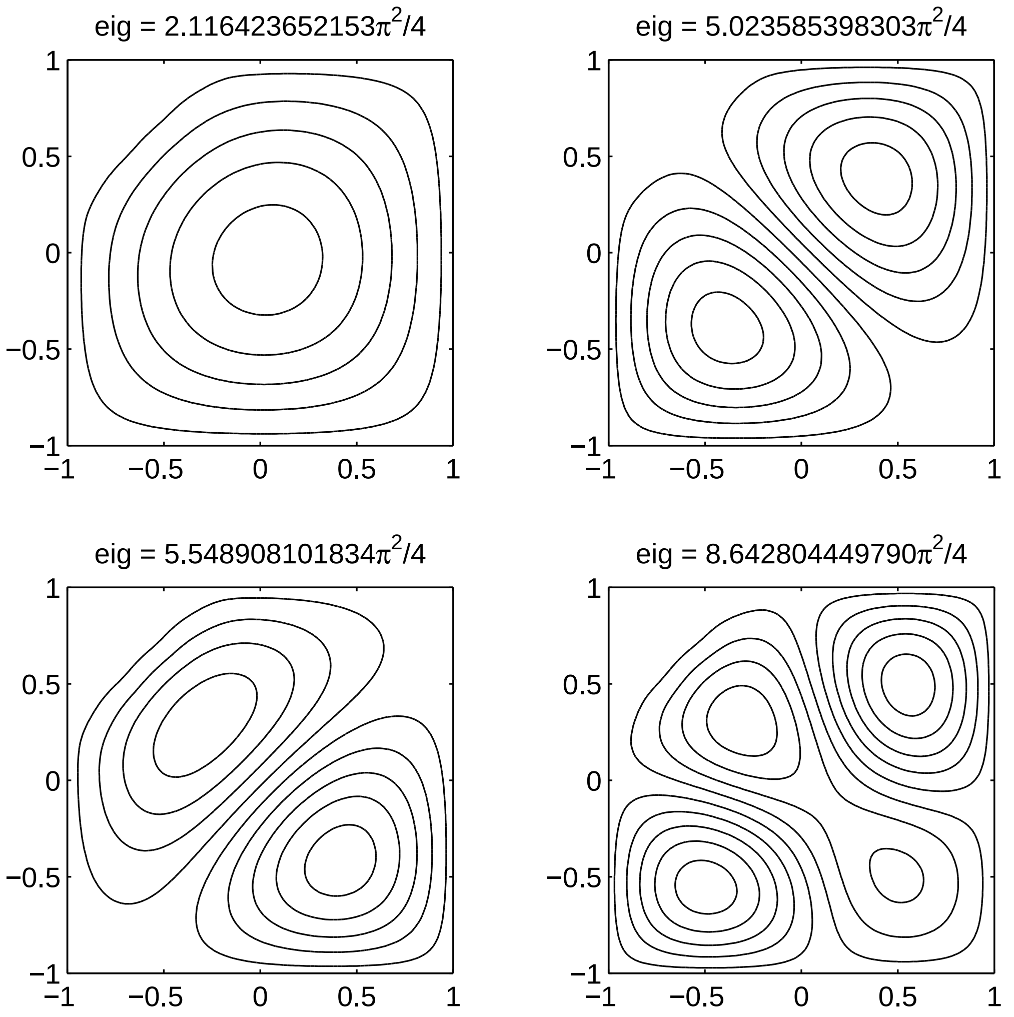
## Output 23a



Output 23a: *First four eigenmodes of the Laplace problem* (9.4) *with* $f(x,y) = 0$. *These plots were produced by running Program* 23 *with the "+ perturbation" line commented out.*

$4.8 \times 10^8$. Thus this perturbation is not small at all in amplitude, though it is limited in extent. It is analogous to the "barrier functions" utilized in the field of optimization of functions with constraints. The effect on the eigenmodes is clear. In Output 23b we see that all four eigenmodes avoid the upper left corner; the values there are very close to zero. It is approximately as if we had solved the eigenvalue problem on the unit square with a corner snipped off. All four eigenvalues have increased, as they must, and the second and third eigenvalues are no longer degenerate. What we find instead is that mode 3, which had low amplitude in the barrier region, has changed a little, whereas

## Output 23b



Output 23b: *First four eigenmodes of the perturbed Laplace problem* (9.4) *with* $f(x, y) = \exp(20(y-x-1))$. *These plots were produced by running Program* 23 *as written.*

mode 2, which had higher amplitude there, has changed quite a lot. These computed eigenvalues, by the way, are not spectrally accurate; the function $f$ varies too fast to be well resolved on this grid. Experiments with various values of $N$ suggest they are accurate to about three or four digits.

All of our examples of eigenvalue problems so far have involved self-adjoint operators, whose eigenvalues are real and whose eigenvectors can be taken to be orthogonal. Our spectral discretizations are not in fact symmetric matrices (they would be, if we used certain Galerkin rather than collocation methods), but they are reasonably close in the sense of having eigenvectors reasonably close to orthogonal so long as the corresponding eigenvalues are distinct. In general, a matrix with a complete set of orthogonal eigenvectors is said to be *normal.* Normal and nearly normal matrices are the ones whose eigenvalue problems are unproblematic, relatively easy both to solve and to interpret physically.

In a certain minority of applications, however, one encounters matrices or operators that are very far from normal in the sense that the eigenvectors, if a complete set exists, are very far from orthogonal—they form an ill-conditioned basis of the vector space under study. In highly nonnormal cases, it may be informative to compute pseudospectra* rather than spectra [Tre97, TTRD93, Wri00]. Suppose that a square matrix $A$ is given and $\|\cdot\|$ is a physically relevant norm. For each $\epsilon > 0$, the *$\epsilon$-pseudospectrum* of $A$ is the subset of the complex plane

$$\Lambda_\epsilon(A) \;=\; \{z \in \mathbb{C}\colon\; \|(zI - A)^{-1}\| \geq \epsilon^{-1}\}. \tag{9.5}$$

(We use the convention $\|(zI - A)^{-1}\| = \infty$ if $z$ is an eigenvalue of $A$.) Alternatively, $\Lambda_\epsilon(A)$ can be characterized by eigenvalues of perturbed matrices:

$$\Lambda_\epsilon(A) \;=\; \{z \in \mathbb{C}\colon\; z \text{ is an eigenvalue of } A + E \text{ for some } E \text{ with } \|E\| \leq \epsilon\}. \tag{9.6}$$

If $\|\cdot\|$ is the 2-norm, as is convenient and physically appropriate in most applications (sometimes after a diagonal similarity transformation to get the scaling right), then a further equivalence is

$$\Lambda_\epsilon(A) = \{z \in \mathbb{C}\colon\; \sigma_{\min}(zI - A) \leq \epsilon\}, \tag{9.7}$$

where $\sigma_{\min}$ denotes the minimum singular value.

Pseudospectra can be computed by spectral methods very effectively, and our final example of this chapter illustrates this. The example returns to the harmonic oscillator (4.6), except that a complex coefficient $c$ is now put in front of the quadratic term. We define our linear operator $L$ by

$$Lu \;=\; -u_{xx} + cx^2u, \qquad x \in \mathbb{R}. \tag{9.8}$$

The eigenvalues and eigenvectors for this problem are readily determined analytically: they are $\sqrt{c}\,(2k+1)$ and $\exp(-c^{1/2}x^2/2)H_k(c^{1/4}x)$ for $k = 0, 1, 2, \ldots$, where $H_k$ is the $k$th Hermite polynomial [Exn83]. However, as E. B. Davies first noted [Dav99], the eigenmodes are exponentially far from orthogonal. Output 24 shows pseudospectra for (9.8) with $c = 1 + 3i$ computed in a

*Pseudospectra (plural of pseudospectrum) are sets in the complex plane; pseudospectral methods are spectral methods based on collocation, i.e., pointwise evaluations rather than integrals. There is no connection—except that pseudospectral methods are very good at computing pseudospectra!

straightforward fashion based on (9.7). We discretize $L$ spectrally, evaluate $\sigma_{\min}(zI - L)$ on a grid of points $z_{ij}$, then send the results to a contour plotter.

For the one and only time in this book, the plot printed as Output 24 is not exactly what would be produced by the corresponding program as listed. Program 24 evaluates $\sigma_{\min}(zI - L)$ on a relatively coarse $26 \times 21$ grid; after 546 complex singular value decompositions, a relatively crude approximation to Output 24 is produced. For the published figure, we made the grid four times finer in each direction by replacing `0:2:50` by `0:.5:50` and `0:2:40` by `0:.5:40`. This slowed down the computation by a factor of 16. (As it happens, alternative algorithms can be used to speed up this calculation of pseudospectra and get approximately that factor of 16 back again; see [Tre99, Wri00].)

One can infer from Output 24 that although the eigenvalues of the complex harmonic oscillator are regularly spaced numbers along a ray in the complex plane, all but the first few of them would be of doubtful physical significance in a physical problem described by this operator. Indeed, the resolvent norm appears to grow exponentially as $|z| \to \infty$ along any ray with argument between 0 and $\arg c$, so that every value of $z$ sufficiently far out in this infinite sector is an $\epsilon$-pseudoeigenvalue for an exponentially small value of $\epsilon$.

We shall see three further examples of eigenvalue calculations later in the book. We summarize the eigenvalue examples ahead by continuing the table displayed at the beginning of this chapter:

| | | |
|---|---|---|
| Program 28: | circular membrane, | polar coordinates; |
| Program 39: | square plate, | clamped boundary conditions; |
| Program 40: | Orr–Sommerfeld operator, | complex arithmetic. |

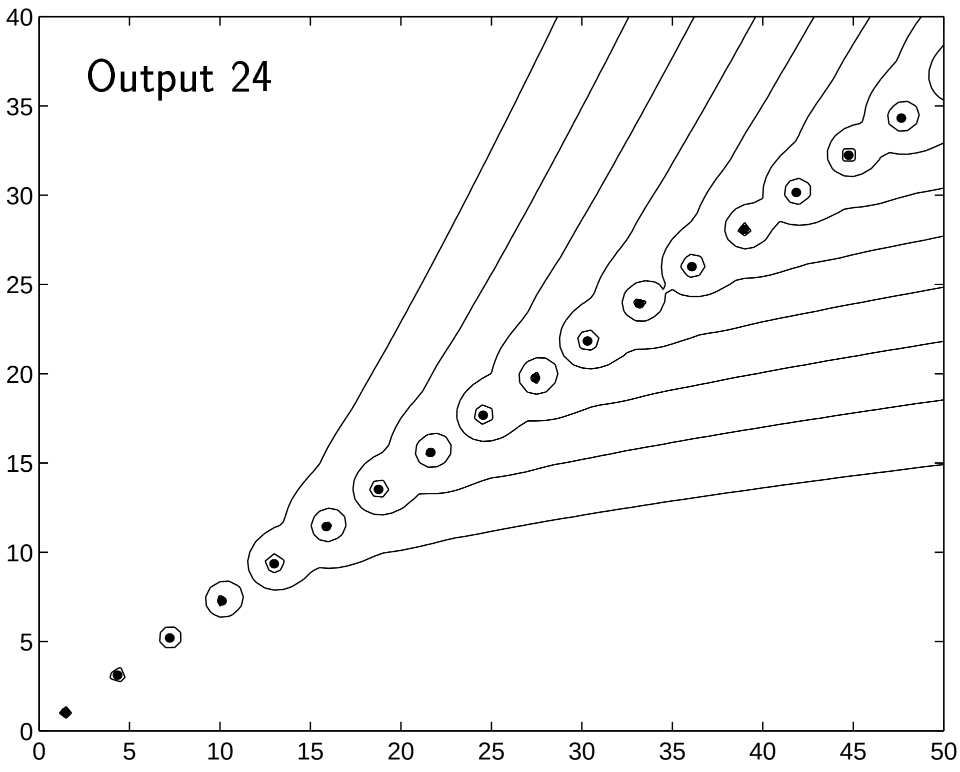
Summary of This Chapter. Spectral discretization can turn eigenvalue and pseudospectra problems for ODEs and PDEs into the corresponding problems for matrices. If the matrix dimension is large, it may be best to solve these by Krylov subspace methods such as the Lanczos or Arnoldi iterations.

## Exercises

**9.1.** Modify Program 23 so that it produces a plot on a log scale of the error in the computed lowest eigenvalue represented in the first panel of Output 23a as a function of $N$. Now let $\tau > 0$ be fixed and let $E_N = \inf_p \|p(x) - \sin(\tau x)\|_\infty$, where $\|f\|_\infty = \sup_{x\in[-1,1]} |f(x)|$, denote the error in degree $N$ minimax polynomial approximation to $\sin(\tau x)$ on $[-1,1]$. It is known (see equation (6.77) of [Mei67]) that for even $N$, as $N \to \infty$, $E_N \sim 2^{-N}\tau^{N+1}/(N+1)!$. Explain which value of $\tau$ should be taken for this result to be used to provide an order of magnitude estimate of the results in the plot. How close is the estimate to the data? (Compare Exercise 5.3.)

## Program 24

```
% p24.m - pseudospectra of Davies's complex harmonic oscillator
%         (For finer, slower plot, change 0:2 to 0:.5.)

% Eigenvalues:
  N = 70; [D,x] = cheb(N); x = x(2:N);
  L = 6; x = L*x; D = D/L;                    % rescale to [-L,L]
  A = -D^2; A = A(2:N,2:N) + (1+3i)*diag(x.^2);
  clf, plot(eig(A),'.','markersize',14)
  axis([0 50 0 40]), drawnow, hold on

% Pseudospectra:
  x = 0:2:50; y = 0:2:40; [xx,yy] = meshgrid(x,y); zz = xx+1i*yy;
  I = eye(N-1); sigmin = zeros(length(y),length(x));
  h = waitbar(0,'please wait...');
  for j = 1:length(x), waitbar(j/length(x))
    for i = 1:length(y), sigmin(i,j) = min(svd(zz(i,j)*I-A)); end
  end, close(h)
  contour(x,y,sigmin,10.^(-4:.5:-.5)), colormap([0 0 0]);
```



Output 24: *Eigenvalues and $\epsilon$-pseudospectra in $\mathbb{C}$ of the complex harmonic oscillator* (9.8), $c = 1 + 3i$, $\epsilon = 10^{-0.5}, 10^{-1}, 10^{-1.5}, \ldots, 10^{-4}$.

**9.2.** $A \otimes (B \otimes C) = (A \otimes B) \otimes C$: True or false?

**9.3.** Modify Program 23 so that it finds the lowest eigenvalue of the Laplacian on the cube $[-1,1]^3$ rather than the square $[-1,1]^2$. For $N = 6$ and 8, how big is the matrix you are working with, how accurate are the results, and how long does the computation take? Estimate what the answers would be for $N = 12$.

**9.4.** In continuation of Exercise 9.3, you can solve the problem with $N = 12$ if you use MATLAB's iterative eigenvalue solver `eigs` rather than its "direct" solver `eig`. Modify your code further to use `eigs`, and be sure that `eigs` is given a sparse matrix to work with (putting `speye` instead of `eye` in your code will ensure this). With $N = 12$, how long does the computation take, and how accurate are the results?

**9.5.** Consider a circular membrane of radius 1 that vibrates according to the second-order wave equation $y_{tt} = r^{-1}(ry_r)_r + r^{-2}y_{\theta\theta}$, $y(1,t) = 0$, written in polar coordinates. Separating variables leads to consideration of solutions $y(r,\theta,t) = u(r)e^{im\theta}e^{i\omega t}$, with $u(r)$ satisfying $r^{-1}(ru_r)_r + (\omega^2 - r^{-2}m^2)u = 0$, $u_r(0) = 0$, $u(1) = 0$. This is a second-order, linear ODE boundary value problem with homogeneous boundary conditions, so one solution is $u(r) = 0$. Nonzero solutions will only occur for eigenvalues $\omega$ of the equation

$$r^{-1}(ru_r)_r - r^{-2}m^2u = -\omega^2 u, \qquad u_r(0) = u(1) = 0. \tag{9.9}$$

This is a form of *Bessel's equation*, and the solutions are Bessel functions $J_m(\omega r)$, where $\omega$ has the property $J_m(\omega) = 0$. Write a MATLAB program based on a spectral method that, for given $m$, constructs a matrix whose smaller eigenvalues approximate the smaller eigenvalues of (9.9). (*Hint.* One method of implementing the Neumann boundary condition at $r = 0$ is described on p. 137.) List the approximations to the first six eigenvalues $\omega$ produced by your program for $m = 0, 1$ and $N = 5, 10, 15, 20$.

**9.6.** In continuation of Exercise 9.5, the first two eigenvalues for $m = 1$ differ nearly, but not quite, by a factor of 2. Suppose, with musical harmony in mind, we wish to design a membrane with radius-dependent physical properties such that these two eigenvalues have ratio exactly 2. Consider the modified boundary value eigenvalue problem

$$r^{-1}(p(r)ru_r)_r - r^{-2}m^2u = -\omega^2 u, \qquad u_r(0) = u(1) = 0,$$

where $p(r) = 1 + \alpha\sin^2(\pi r)$ for some real number $\alpha$. Produce a plot that shows the first eigenvalue and $\frac{1}{2}$ times the second eigenvalue as functions of $\alpha$. For what value of $\alpha$ do the two curves intersect? By solving numerically an appropriate nonlinear equation, determine this critical value of $\alpha$ to at least six digits. Can you explain why a correction of the form $\alpha p(r)$ modifies the ratio of the eigenvalues in the direction required?

**9.7.** Exercise 6.8 (p. 59) considered powers of the Chebyshev differentiation matrix $D_N$. For $N = 20$, produce a plot of the eigenvalues and $\epsilon$-pseudospectra of $D_N$ for $\epsilon = 10^{-2}, 10^{-3}, \ldots, 10^{-16}$. Comment on how this plot relates to the results of that exercise.

**9.8.** Download the MATLAB programs from [Wri00] for computing pseudospectra and use them to generate a figure similar to Output 24. How does the computation time compare to that of Program 24?