Mathematics Monoqraph Series 3

Jie Shen Tao Tang

Spectral and High-Order Methods with Applications

SCIENCE PRESS
Beijjng

Responsible Editor: Chen Yuzhuo

Published by Science Press
16 Donghuangchenggen North Street
Beijing 100717, China

Printed in Beijing

ISBN 7-03-017722-3/0.2553(Beijing)

Preface

This book expands lecture notes by the authors for a course on *Introduction of Spectral Methods* taught in the past few years at Penn State University, Simon Fraser University, the Chinese University of Hong Kong, Hong Kong Baptist University, Purdue University and the Chinese Academy of Sciences. Our lecture notes were also used by Prof. Zhenhuan Teng in his graduate course at Peking University.

The overall emphasis of the present book is to present some basic spectral and high-order algorithms together with applications to some linear and nonlinear problems that one frequently encounters in practice. The algorithms in the book are presented in a *pseudocode* format or with MATLAB or FORTRAN codes that contain additional details beyond the mathematical formulas. The reader can easily write computer routines based on the pseudocodes in any standard computer language. We believe that the readers learn and understand numerical methods best by seeing how algorithms are developed from the mathematical theory and then writing and testing computer implementations of them. For those interested in the numerical analysis of the spectral methods, we have also provided self-contained error analysis for some basic spectral-Galerkin algorithms presented in the book. Our aim is to provide a sufficient background on the implementation and analysis of spectral and high-order methods so that the readers can approach the current research literature with the necessary tools and understanding.

We hope that this book will be useful for people studying spectral methods on their own. It may also serve as a textbook for advanced undergraduate/beginning graduate students. The only prerequisite for the present book is a standard course in Numerical Analysis.

This project has been supported by NSERC Canada, National Science Foundation, Research Grant Council of Hong Kong, and International Research Team of Complex System of the Chinese Academy of Sciences. In writing this book, we have received much help from our friends and students. In particular, we would like to thank Dr. Lilian Wang of Nanyang Technical University of Singapore for his many contributions throughout the book. We are grateful to the help provided by Zhongzhi Bai of the Chinese Academy of Sciences, Weizhu Bao of National University of Singapore, Raymond Chan of Chinese University of Hong Kong, Wai Son Don of Brown

University, Heping Ma of Shanghai University and Xuecheng Tai of Bergen University of Norway. Our gratitude also goes to Professor Hermann Brunner of Memorial University of Newfoundland, Dr. Zhengru Zhang of Beijing Normal University, and the following graduate students at Purdue, Qirong Fang, Yuen-Yick Kwan, Hua Lin, Xiaofeng Yang and Yanhong Zhao, who have read the entire manuscripts and provided many constructive suggestions. Last but not the least, we would like to thank our wives and children for their love and support.

A website relevant to this book can be found in

`http://www.math.hkbu.edu.hk/~ttang/PGteaching` or

`http://lsec.cc.ac.cn/~ttang/PGteaching`

We welcome comments and corrections to the book. We can be reached by email to

`shen@math.purdue.edu` (Shen) and `ttang@math.hkbu.edu.hk` (Tang).

Jie Shen
Purdue University

Tao Tang
Hong Kong Baptist University

Contents

Preface

Chapter 1 Preliminaries 1

1.1 Some basic ideas of spectral methods 2

1.2 Orthogonal polynomials 6

1.3 Chebyshev and Legendre polynomials 15

1.4 Jacobi polynomials and generalized Jacobi polynomials 23

1.5 Fast Fourier transform 27

1.6 Several popular time discretization methods 38

1.7 Iterative methods and preconditioning 48

1.8 Error estimates of polynomial approximations 61

Chapter 2 Spectral-Collocation Methods 68

2.1 Differentiation matrices for polynomial basis functions 69

2.2 Differentiation matrices for Fourier collocation methods 79

2.3 Eigenvalues of Chebyshev collocation operators 84

2.4 Chebyshev collocation method for two-point BVPs 91

2.5 Collocation method in the weak form and preconditioning 99

Chapter 3 Spectral-Galerkin Methods 105

3.1 General setup 105

3.2 Legendre-Galerkin method 109

3.3 Chebyshev-Galerkin method 114

3.4 Chebyshev-Legendre Galerkin method 118

3.5 Preconditioned iterative method 121

3.6 Spectral-Galerkin methods for higher-order equations 126

3.7 Error estimates 131

Chapter 4 Spectral Methods in Unbounded Domains 143
4.1 Hermite spectral methods 144
4.2 Laguerre spectral methods 158
4.3 Spectral methods using rational functions 170
4.4 Error estimates in unbounded domains 177
Chapter 5 Some applications in one space dimension 183
5.1 Pseudospectral methods for boundary layer problems 184
5.2 Pseudospectral methods for Fredholm integral equations 190
5.3 Chebyshev spectral methods for parabolic equations 196
5.4 Fourier spectral methods for the KdV equation 204
5.5 Fourier method and filters 214
5.6 Essentially non-oscillatory spectral schemes 222
Chapter 6 Spectral methods in Multi-dimensional Domains 231
6.1 Spectral-collocation methods in rectangular domains 233
6.2 Spectral-Galerkin methods in rectangular domains 237
6.3 Spectral-Galerkin methods in cylindrical domains 243
6.4 A fast Poisson Solver using finite differences 247
Chapter 7 Some applications in multi-dimensions 256
7.1 Spectral methods for wave equations 257
7.2 Laguerre-Hermite method for Schrödinger equations 264
7.3 Spectral approximation of the Stokes equations 276
7.4 Spectral-projection method for Navier-Stokes equations 282
7.5 Axisymmetric flows in a cylinder 288
Appendix A Some online software 299
A.1 MATLAB Differentiation Matrix Suite 300
A.2 PseudoPack 308
Bibliography 313
Index 323

Chapter 1

Preliminaries

Contents

1.1 Some basic ideas of spectral methods 2
1.2 Orthogonal polynomials 6
1.3 Chebyshev and Legendre polynomials 15
1.4 Jacobi polynomials and generalized Jacobi polynomials 23
1.5 Fast Fourier transform 27
1.6 Several popular time discretization methods 38
1.7 Iterative methods and preconditioning 48
1.8 Error estimates of polynomial approximations 61

In this chapter, we present some preliminary materials which will be used throughout the book. The first section set the stage for the introduction of spectral methods. In Sections 1.2∼1.4, we present some basic properties of orthogonal polynomials, which play an essential role in spectral methods, and introduce the notion of generalized Jacobi polynomials. Since much of the success and popularity of spectral methods can be attributed to the invention of Fast Fourier Transform (FFT), an algorithmic description of the FFT is presented in Section 1.5. In the next two sections, we collect some popular time discretization schemes and iterative schemes which will be frequently used in the book. In the last section, we present a concise error analysis for several projection operators which serves as the basic ingredients for the error analysis of spectral methods.

1.1 Some basic ideas of spectral methods

Comparison with the finite element method
Computational efficiency
Fourier spectral method
Phase error

Finite Difference (FD) methods approximate derivatives of a function by *local* arguments (such as $u'(x) \approx (u(x+h)-u(x-h))/2h$, where h is a small grid spacing) - these methods are typically designed to be exact for polynomials of low orders. This approach is very reasonable: since the derivative is a local property of a function, it makes little sense (and is costly) to invoke many function values far away from the point of interest.

In contrast, spectral methods are *global*. The traditional way to introduce them starts by approximating the function as a sum of very smooth basis functions:

$$u(x) \approx \sum_{k=0}^{N} a_k \Phi_k(x),$$

where the $\Phi_k(x)$ are polynomials or trigonometric functions. In practice, there are many feasible choices of the basis functions, such as:

$\Phi_k(x) = e^{ikx}$ (the Fourier spectral method); $\Phi_k(x) = T_k(x)$ ($T_k(x)$ are the Chebyshev polynomials; the Chebyshev spectral method); $\Phi_k(x) = L_k(x)$ ($L_k(x)$ are the Legendre polynomials; the Legendre spectral method).

In this section, we will describe some basic ideas of spectral methods. For ease of exposition, we consider the Fourier spectral method (i.e. the basis functions are chosen as e^{ikx}). We begin with the periodic heat equation, starting at time 0 from $u_0(x)$:

$$u_t = u_{xx}, \tag{1.1.1}$$

with a periodic boundary condition $u(x,0) = u_0(x) = u_0(x+2\pi)$. Since the exact solution u is periodic, it can be written as an infinite Fourier series. The approximate solution u^N can be expressed as a *finite* series. It is

$$u^N(x,t) = \sum_{k=0}^{N-1} a_k(t) e^{ikx}, \qquad x \in [0, 2\pi),$$

where each $a_k(t)$ is to be determined.

Comparison with the finite element method

We may compare the spectral method (before actually describing it) to the finite element method. One difference is this: the trial functions τ_k in the finite element method are usually 1 at the mesh-point, $x_k = kh$ with $h = 2\pi/N$, and 0 at the other mesh-points, whereas e^{ikx} is nonzero everywhere. That is not such an important distinction. We could produce from the exponentials an interpolating function like τ_k, which is zero at all mesh-points except at $x = x_k$:

$$F_k(x) = \frac{1}{N}\sin\frac{N}{2}(x - x_k)\cot\frac{1}{2}(x - x_k), \qquad N \text{ even}, \tag{1.1.2}$$

$$F_k(x) = \frac{1}{N}\sin\frac{N}{2}(x - x_k)\csc\frac{1}{2}(x - x_k), \qquad N \text{ odd}. \tag{1.1.3}$$

Of course it is not a piecewise polynomial; that distinction is genuine. A consequence of this difference is the following:

> Each function F_k spreads over the whole solution interval, whereas τ_k is zero in all elements not containing x_k. The stiffness matrix is sparse for the finite element method; in the spectral method it is full.

The computational efficiency

Since the matrix associated with the spectral method is full, the spectral method seems more time-consuming than finite differences or finite elements. In fact, the spectral method had not been used widely for a long time. The main reason is the expensive cost in computational time. However, the discovery of the Fast Fourier Transform (FFT) by Cooley and Tukey[33] solves this problem. We will describe the Cooley-Tukey algorithm in Chapter 5. The main idea is the following. Let $w_N = e^{2\pi i/N}$ and

$$(\mathcal{F}_N)_{jk} = w_N^{jk} = \cos\frac{2\pi jk}{N} + i\sin\frac{2\pi jk}{N}, \qquad 0 \leqslant j, \quad k \leqslant N-1.$$

Then for any N-dimensional vector v_N, the usual N^2 operations in computing $\mathcal{F}_N v_N$ are reduced to $N\log_2 N$. The significant improvement can be seen from the following table:

N	N^2	$N\log_2 N$	N	N^2	$N\log_2 N$
16	256	64	256	65536	2048

```
 32     1024       160        512     262144       4608
 64     4096       384       1024    1048576      10240
128    16384       896       2048    4194304      22528
```

The Fourier spectral method

Unlike finite differences or finite elements, which replace the right-hand side u_{xx} by differences at nodes, the spectral method uses u_{xx}^N exactly. In the spectral method, there is no Δx. The derivatives with respect to space variables are computed explicitly and correctly.

The Fourier approximation u^N is a combination of oscillations e^{ikx} up to frequency $N-1$, and we simply differentiate them; hence

$$u_t^N = u_{xx}^N$$

becomes

$$\sum_{k=0}^{N-1} a_k'(t)e^{ikx} = \sum_{k=0}^{N-1} a_k(t)(ik)^2 e^{ikx}.$$

Since frequencies are uncoupled, we have $a_k'(t) = -k^2 a_k(t)$, which gives

$$a_k(t) = e^{-k^2 t} a_k(0),$$

where the values $a_k(0)$ are determined by using the initial function:

$$a_k(0) = \frac{1}{2\pi}\int_0^{2\pi} u_0(x)e^{-ikx}\mathrm{d}x.$$

It is an easy matter to show that

$$|u(x,t) - u^N(x,t)| = \left|\sum_{k=N}^{\infty} a_k(0)e^{ikx}e^{-k^2 t}\right|$$

$$\leqslant \max_k |a_k(0)| \sum_{k=N}^{\infty} e^{-k^2 t}$$

$$\leqslant \max_{0\leqslant x\leqslant 2\pi} |u_0(x)| \int_N^{\infty} e^{-tx^2}\mathrm{d}x.$$

Therefore, the error goes to zero very rapidly as N becomes reasonably large. The

convergence rate is determined by the integral term

$$J(t,N) := \int_N^\infty e^{-tx^2}\,\mathrm{d}x = \sqrt{\frac{\pi}{4t}}\,\mathrm{erfc}(\sqrt{t}N),$$

where `erfc`(x) is the complementary error function (both FORTRAN and MATLAB have this function). The following table lists the value of $J(t,N)$ at several values of t:

N	J(0.1, N)	J(0.5, N)	J(1, N)
1	1.8349e+00	3.9769e-01	1.3940e-01
2	1.0400e+00	5.7026e-02	4.1455e-03
3	5.0364e-01	3.3837e-03	1.9577e-05
4	2.0637e-01	7.9388e-05	1.3663e-08
5	7.1036e-02	7.1853e-07	1.3625e-12
6	2.0431e-02	2.4730e-09	1.9071e-17
7	4.8907e-03	3.2080e-12	3.7078e-23
8	9.7140e-04	1.5594e-15	9.9473e-30

In more general problems, the equation in time will not be solved exactly. It needs a difference method with time step Δt, as Chapter 5 will describe. For derivatives with respect to space variables, there are two ways:

(1) Stay with the harmonics e^{ikx} or $\sin kx$ or $\cos kx$, and use FFT to go between coefficients a_k and mesh values $u^N(x_j,t)$. Only the mesh values enter the difference equation in time.

(2) Use an expansion $U = \sum U_k(t)F_k(x)$, where $F_k(x)$ is given by (1.1.2) and (1.1.3), that works directly with values U_k at mesh points (where $F_k = 1$). There is a *differentiation matrix* D that gives mesh values of the derivatives, $D_{jk} = F_k'(x_j)$. Then the approximate heat equation becomes $U_t = D^2U$.

Phase error

The fact that x-derivatives are exact makes spectral methods free of phase error. Differentiation of the multipliers e^{jkx} give the right factor ik while finite differences lead to the approximate factor iK:

$$\frac{e^{ik(x+h)} - e^{ik(x-h)}}{2h} = iKe^{ikx}, \qquad K = \frac{\sin kh}{h}\,.$$

When kh is small and there are enough mesh points in a wavelength, K is close to k. When kh is large, K is significantly smaller than k. In the case of the heat

equation (1.1.1) it means a slower wave velocity. For details, we refer to Richtmyer and Morton[131] and LeVeque [101]. In contrast, the spectral method can follow even the nonlinear wave interactions that lead to turbulence. In the context of solving high Reynolds number flow, the low physical dissipation will not be overwhelmed by large numerical dissipation.

Exercise 1.1

Problem 1 Consider the linear heat equation (1.1.1) with homogeneous Dirichlet boundary conditions $u(-1,t) = 0$ and $u(1,t) = 0$. If the initial condition is $u(x,0) = \sin(\pi x)$, then the exact solution of this problem is given by $u(x,t) = e^{-\pi^2 t}\sin(\pi x)$. It has the infinite Chebyshev expansion

$$u(x,t) = \sum_{n=0}^{\infty} b_n(t) T_n(x),$$

where

$$b_n(t) = \frac{1}{c_n} J_n(\pi) e^{-\pi^2 t},$$

with $c_0 = 2$ and $c_n = 1$ if $n \geqslant 1$.

a. Calculate

$$J_n(\pi) = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} T_n(x)\sin(\pi x)\mathrm{d}x$$

by some numerical method (e.g. Simpson's rule)①;

b. Plot $J_n(\pi)$ against n for $n \leqslant 25$. This will show that the truncation series converges at an exponential rate (a well-designed collocation method will do the same).

1.2 Orthogonal polynomials

Existence
Zeros of orthogonal polynomials
Polynomial interpolations
Quadrature formulas
Discrete inner product and discrete transform

① Hint: (a) Notice that $J_n(\pi) = 0$ when n is even; (b) a coordinate transformation like $x = \cos\theta$ may be used.

Orthogonal polynomials play a fundamental role in the implementation and analysis of spectral methods. It is thus essential to understand some general properties of orthogonal polynomials. Two functions f and g are said to be *orthogonal* in the weighted Sobolev space $L^2_\omega(a,b)$ if

$$\langle f,\, g\rangle := (f,g)_\omega := \int_a^b \omega(x) f(x) g(x) \mathrm{d}x = 0,$$

where ω is a fixed positive *weight function* in (a,b). It can be easily verified that $\langle \cdot,\, \cdot \rangle$ defined above is an inner product in $L^2_\omega(a,b)$.

A sequence of *orthogonal polynomials* is a sequence $\{p_n\}_{n=0}^\infty$ of polynomials with $\deg(p_n) = n$ such that

$$\langle p_i,\, p_j\rangle = 0 \qquad \text{for} \quad i \neq j. \tag{1.2.1}$$

Since orthogonality is not altered by multiplying a nonzero constant, we may normalize the polynomial p_n so that the coefficient of x^n is one, i.e.,

$$p_n(x) = x^n + a_{n-1}^{(n)} x^{n-1} + \cdots + a_0^{(n)}.$$

Such a polynomial is said to be *monic*.

Existence

Our immediate goal is to establish the existence of orthogonal polynomials. Although we could, in principle, determine the coefficients $a_j^{(n)}$ of p_n in the natural basis $\{x^j\}$ by using the orthogonality conditions (1.2.1), it is more convenient, and numerically more stable, to express p_{n+1} in terms of lower-order orthogonal polynomials. To this end, we need the following general result:

Let $\{p_n\}_{n=0}^\infty$ be a sequence of polynomials such that p_n is exactly of degree n. If

$$q(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0, \tag{1.2.2}$$

then q can be written uniquely in the form

$$q(x) = b_n p_n + b_{n-1} p_{n-1} + \cdots + b_0 p_0. \tag{1.2.3}$$

In establishing this result, we may assume that the polynomials $\{p_n\}$ are monic. We shall prove this result by induction. For $n = 0$, we have

$$q(x) = a_0 = a_0 \cdot 1 = a_0 p_0(x).$$

Hence we must have $b_0 = a_0$. Now assume that q has the form (1.2.2). Since p_n is the only polynomial in the sequence $p_n, p_{n-1}, \cdots, p_0$ that contains x^n and since p_n is monic, it follows that we must have $b_n = a_n$. Hence, the polynomial $q - a_n p_n$ is of degree $n-1$. Thus, by the induction hypothesis, it can be expressed uniquely in the form

$$q - a_n p_n = b_{n-1} p_{n-1} + \cdots + b_0 p_0,$$

which establishes the result.

A consequence of this result is the following:

Lemma 1.2.1 *If the sequence of polynomials $\{p_n\}_{n=0}^{\infty}$ is monic and orthogonal, then the polynomial p_{n+1} is orthogonal to any polynomial q of degree n or less.*

We can establish this by the following observation:

$$\langle p_{n+1},\, q\rangle = b_n\langle p_{n+1},\, p_n\rangle + b_{n-1}\langle p_{n+1},\, p_{n-1}\rangle + \cdots + b_0\langle p_{n+1},\, p_0\rangle = 0,$$

where the last equality follows from the orthogonality of the polynomials $\{p_n\}$.

We now prove the existence of orthogonal polynomials①. Since p_0 is monic and of degree zero, we have

$$p_0(x) \equiv 1.$$

Since p_1 is monic and of degree one, it must have the form

$$p_1(x) = x - \alpha_1.$$

To determine α_1, we use orthogonality:

$$0 = \langle p_1,\, p_0\rangle = \int_a^b \omega(x) x \mathrm{d}x - \alpha_1 \int_a^b \omega(x)\mathrm{d}x.$$

Since the weight function is positive in (a, b), it follows that

$$\alpha_1 = \int_a^b \omega(x) x \mathrm{d}x \Big/ \int_a^b \omega(x)\mathrm{d}x.$$

In general we seek p_{n+1} in the form $p_{n+1} = xp_n - \alpha_{n+1}p_n - \beta_{n+1}p_{n-1} - \gamma_{n+1}p_{n-2} - \cdots$. As in the construction of p_1, we use orthogonality to determine the coefficients above. To determine α_{n+1}, write

$$0 = \langle p_{n+1},\, p_n\rangle = \langle xp_n,\, p_n\rangle - \alpha_{n+1}\langle p_n,\, p_n\rangle - \beta_{n+1}\langle p_{n-1},\, p_n\rangle - \cdots .$$

① The procedure described here is known as Gram-Schmidt orthogonalization.

By orthogonality, we have

$$\int_a^b x\omega p_n^2 \mathrm{d}x - \alpha_{n+1}\int_a^b \omega p_n^2 \mathrm{d}x = 0,$$

which yields

$$\alpha_{n+1} = \int_a^b x\omega p_n^2 \mathrm{d}x \Big/ \int_a^b \omega p_n^2 \mathrm{d}x.$$

For β_{n+1}, using the fact $\langle p_{n+1},\, p_{n-1}\rangle = 0$ gives

$$\beta_{n+1} = \int_a^b x\omega p_n p_{n-1} \mathrm{d}x \Big/ \int_a^b \omega p_{n-1}^2 \mathrm{d}x.$$

The formulas for the remaining coefficients are similar to the formula for β_{k+1}; e.g.

$$\gamma_{n+1} = \int_a^b x\omega p_n p_{n-2} \mathrm{d}x \Big/ \int_a^b \omega p_{n-2}^2 \mathrm{d}x.$$

However, there is a surprise here. The numerator $\langle xp_n,\, p_{n-2}\rangle$ can be written in the form $\langle p_n,\, xp_{n-2}\rangle$. Since xp_{n-2} is of degree $n-1$ it is orthogonal to p_n. Hence $\gamma_{n+1} = 0$, and likewise the coefficients of p_{n-3}, p_{n-4}, etc. are all zeros.

To summarize:

The orthogonal polynomials can be generated by the following recurrence:

$$\begin{cases} p_0 = 1, \\ p_1 = x - \alpha_1, \\ \quad \cdots\cdots \\ p_{n+1} = (x - \alpha_{n+1})p_n - \beta_{n+1}p_{n-1}, \quad n \geqslant 1, \end{cases} \tag{1.2.4}$$

where

$$\alpha_{n+1} = \int_a^b x\omega p_n^2 \mathrm{d}x \Big/ \int_a^b \omega p_n^2 \mathrm{d}x \text{ and } \beta_{n+1} = \int_a^b x\omega p_n p_{n-1} \mathrm{d}x \Big/ \int_a^b \omega p_{n-1}^2 \mathrm{d}x.$$

The first two equations in the recurrence merely start things off. The right-hand side of the third equation contains three terms and for that reason is called the *three-term recurrence relation* for the orthogonal polynomials.

Zeros of orthogonal polynomials

The zeros of the orthogonal polynomials play a particularly important role in the implementation of spectral methods.

Lemma 1.2.2 *The zeros of p_{n+1} are real, simple, and lie in the open interval (a, b).*

The proof of this lemma is left as an exercise. Moreover, one can derive from the three term recurrence relation (1.2.4) the following useful result.

Theorem 1.2.1 *The zeros $\{x_j\}_{j=0}^{n}$ of the orthogonal polynomial p_{n+1} are the eigenvalues of the symmetric tridiagonal matrix*

$$A_{n+1} = \begin{bmatrix} \alpha_0 & \sqrt{\beta_1} & & & \\ \sqrt{\beta_1} & \alpha_1 & \sqrt{\beta_2} & & \\ & \ddots & \ddots & \ddots & \\ & & \sqrt{\beta_{n-1}} & \alpha_{n-1} & \sqrt{\beta_n} \\ & & & \sqrt{\beta_n} & \alpha_n \end{bmatrix}, \tag{1.2.5}$$

where

$$\alpha_j = \frac{b_j}{a_j}, \ \text{for} \ j \geqslant 0; \quad \beta_j = \frac{c_j}{a_{j-1}a_j}, \ \text{for} \ j \geqslant 1, \tag{1.2.6}$$

with $\{a_k, b_k, c_k\}$ being the coefficients of the three term recurrence relation (cf. (1.2.4)*) written in the form:*

$$p_{k+1} = (a_k x - b_k)p_k - c_k p_{k-1}, \qquad k \geqslant 0. \tag{1.2.7}$$

Proof The proof is based on introducing

$$\tilde{p}_n(x) = \frac{1}{\sqrt{\gamma_n}} p_n(x),$$

where γ_n is defined by

$$\gamma_n = \frac{c_n a_{n-1}}{a_n}\gamma_{n-1}, \ \ n \geqslant 1, \quad \gamma_0 = 1. \tag{1.2.8}$$

We deduce from (1.2.7) that

$$x\tilde{p}_j = \frac{c_j}{a_j}\sqrt{\frac{\gamma_{j-1}}{\gamma_j}}\tilde{p}_{j-1} + \frac{b_j}{a_j}\tilde{p}_j + \frac{1}{a_j}\sqrt{\frac{\gamma_{j+1}}{\gamma_j}}\tilde{p}_{j+1}, \quad j \geqslant 0, \tag{1.2.9}$$

with $\tilde{p}_{-1} = 0$. Owing to (1.2.6) and (1.2.8), it can be rewritten as

$$x\tilde{p}_j(x) = \sqrt{\beta_j}\tilde{p}_{j-1}(x) + \alpha_j\tilde{p}_j(x) + \sqrt{\beta_{j+1}}\tilde{p}_{j+1}(x), \quad j \geqslant 0. \tag{1.2.10}$$

We now take $j = 0, 1, \cdots, n$ to form a system

$$x\widetilde{\mathbf{P}}(x) = A_{n+1}\widetilde{\mathbf{P}}(x) + \sqrt{\beta_{n+1}}\tilde{p}_{n+1}(x)\mathbf{E}_n, \tag{1.2.11}$$

where $\widetilde{\mathbf{P}}(x) = (\tilde{p}_0(x), \tilde{p}_1(x), \cdots, \tilde{p}_n(x))^{\mathrm{T}}$ and $\mathbf{E}_n = (0, 0, \cdots, 0, 1)^{\mathrm{T}}$. Since $\tilde{p}_{n+1}(x_j) = 0,\ 0 \leqslant j \leqslant n$, the equation (1.2.11) at $x = x_j$ becomes

$$x_j\widetilde{\mathbf{P}}(x_j) = A_{n+1}\widetilde{\mathbf{P}}(x_j), \quad 0 \leqslant j \leqslant n. \tag{1.2.12}$$

Hence, the zeros $\{x_j\}_{j=0}^n$ are the eigenvalues of the symmetric tridiagonal matrix A_{n+1}. □

Polynomial interpolations

Let us denote

$$P_N = \{\text{polynomials of degree not exceeding } N\}. \tag{1.2.13}$$

Given a set of points $a = x_0 < x_1 \cdots < x_N = b$ (we usually take $\{x_i\}$ to be zeros of certain orthogonal polynomials), we define the polynomial interpolation operator, $I_N : C(a, b) \to P_N$, associated with $\{x_i\}$, by

$$I_N u(x_j) = u(x_j), \qquad j = 0, 1, \cdots, N. \tag{1.2.14}$$

The following result describes the discrepancy between a function u and its polynomial interpolant $I_N u$. This is a standard result and its proof can be found in most numerical analysis textbook.

Lemma 1.2.3 *If $x_0, x_1, \cdots, x_N$ are distinct numbers in the interval $[a, b]$ and $u \in C^{N+1}[a, b]$, then, for each $x \in [a, b]$, there exists a number ζ in (a, b) such that*

$$u(x) - I_N u(x) = \frac{u^{(N+1)}(\zeta)}{(N+1)!}\prod_{k=0}^{N}(x - x_k), \tag{1.2.15}$$

where $I_N u$ is the interpolating polynomial satisfying (1.2.14).

It is well known that for an arbitrary set of $\{x_j\}$, in particular if $\{x_j\}$ are equally spaced in $[a, b]$, the error in the maximum norm, $\max_{x\in[a,b]} |u(x) - I_N(x)|$, may

not converge as $N \to +\infty$ even if $u \in C^{\infty}[a,b]$. A famous example is the Runge function

$$f(x) = \frac{1}{25x^2+1}, \quad x \in [-1,1], \tag{1.2.16}$$

see Figure 1.1.

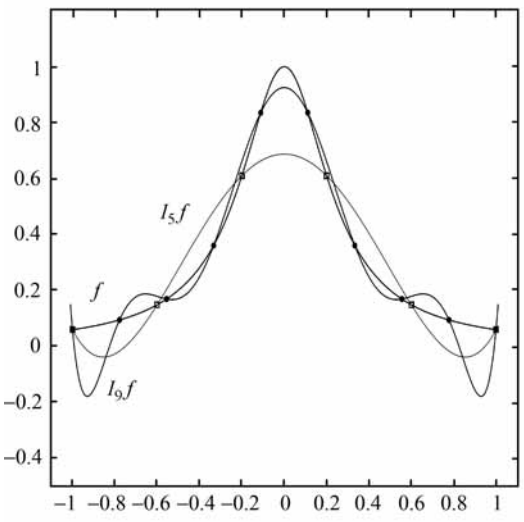

Figure 1.1 Runge function f and the equidistant interpolations $I_5 f$ and $I_9 f$ for (1.2.16)

The approximation gets worse as the number of interpolation points increases.

Hence, it is important to choose a suitable set of points for interpolation. Good candidates are the zeros of certain orthogonal polynomials which are Gauss-type quadrature points, as shown below.

Quadrature formulas

We wish to create quadrature formulas of the type

$$\int_a^b f(x)\omega(x)\mathrm{d}x \approx \sum_{n=0}^{N} A_n f(\gamma_n).$$

If the choice of nodes $\gamma_0, \gamma_1, \cdots, \gamma_n$ is made *a priori*, then in general the above formula is exact for polynomials of degree $\leqslant N$. However, if we are free to choose the nodes γ_n, we can expect quadrature formulas of the above form be exact for polynomials of degree up to $2N+1$.

There are three commonly used quadrature formulas. Each of them is associated

with a set of collocation points which are zeroes of a certain orthogonal polynomial. The first is the well-known Gauss quadrature which can be found in any elementary numerical analysis textbook.

`Gauss Quadrature` Let $x_0, x_1, \cdots, x_N$ be the zeroes of p_{N+1}. Then, the linear system

$$\sum_{j=0}^{N} p_k(x_j)\omega_j = \int_a^b p_k(x)\omega(x)\mathrm{d}x, \qquad 0 \leqslant k \leqslant N, \tag{1.2.17}$$

admits a unique solution $(\omega_0, \omega_1, \cdots, \omega_N)^t$, with $\omega_j > 0$ for $j = 0, 1, \cdots, N$. Furthermore,

$$\sum_{j=0}^{N} p(x_j)\omega_j = \int_a^b p(x)\omega(x)\mathrm{d}x, \quad \text{for all} \quad p \in P_{2N+1}. \tag{1.2.18}$$

The Gauss quadrature is the most accurate in the sense that it is impossible to find x_j, ω_j such that (1.2.18) holds for all polynomials $p \in P_{2N+2}$. However, by Lemma 1.2.1 this set of collocation points $\{x_i\}$ does not include the endpoint a or b, so it may cause difficulties for boundary value problems.

The second is the Gauss-Radau quadrature which is associated with the roots of the polynomial

$$q(x) = p_{N+1}(x) + \alpha p_N(x), \tag{1.2.19}$$

where α is a constant such that $q(a) = 0$. It can be easily verified that $q(x)/(x-a)$ is orthogonal to all polynomials of degree less than or equal to $N-1$ in $L^2_{\tilde{\omega}}(a, b)$ with $\tilde{\omega}(x) = \omega(x)(x-a)$. Hence, the N roots of $q(x)/(x-a)$ are all real, simple and lie in (a, b).

`Gauss-Radau Quadrature` Let $x_0 = a$ and $x_1, \cdots, x_N$ be the zeroes of $q(x)/(x-a)$, where $q(x)$ is defined by (1.2.19). Then, the linear system (1.2.17) admits a unique solution $(\omega_0, \omega_1, \cdots, \omega_N)^t$ with $\omega_j > 0$ for $j = 0, 1, \cdots, N$. Furthermore,

$$\sum_{j=0}^{N} p(x_j)\omega_j = \int_a^b p(x)\omega(x)\mathrm{d}x, \quad \text{for all} \quad p \in P_{2N}. \tag{1.2.20}$$

Similarly, one can construct a Gauss-Radau quadrature by fixing $x_N = b$. Thus, the Gauss-Radau quadrature is suitable for problems with one boundary point.

The third is the Gauss-Lobatto quadrature which is the most commonly used in

spectral approximations since the set of collocation points includes the two endpoints. Here, we consider the polynomial

$$q(x) = p_{N+1}(x) + \alpha p_N(x) + \beta p_{N-1}(x), \tag{1.2.21}$$

where α and β are chosen so that $q(a) = q(b) = 0$. One can verify that $q(x)/((x-a)(x-b))$ is orthogonal to all polynomials of degree less than or equal to $N-2$ in $L^2_{\hat{\omega}}(a,b)$ with $\hat{\omega}(x) = \omega(x)(x-a)(x-b)$. Hence, the $N-1$ zeroes of $q(x)/((x-a)(x-b))$ are all real, simple and lie in (a,b).

`Gauss-Lobatto Quadrature` Let $x_0 = a$, $x_N = b$ and $x_1, \cdots, x_{N-1}$ be the $(N-1)$-roots of $q(x)/((x-a)(x-b))$, where $q(x)$ is defined by (1.2.21). Then, the linear system (1.2.17) admits a unique solution $(\omega_0, \omega_1, \cdots, \omega_N)^t$, with $\omega_j > 0$, for $j = 0, 1, \cdots, N$. Furthermore,

$$\sum_{j=0}^{N} p(x_j)\omega_j = \int_a^b p(x)\omega(x)\mathrm{d}x, \quad \text{for all} \quad p \in P_{2N-1}. \tag{1.2.22}$$

Discrete inner product and discrete transform

For any of the Gauss-type quadratures defined above with the points and weights $\{x_j, \omega_j\}_{j=0}^N$, we can define a discrete inner product in $C[a,b]$ and its associated norm by:

$$(u,v)_{N,\omega} = \sum_{j=0}^{N} u(x_j)v(x_j)\omega_j, \ \|u\|_{N,\omega} = (u,u)_{N,\omega}^{\frac{1}{2}}, \tag{1.2.23}$$

and for $u \in C[a,b]$, we can write

$$u(x_j) = I_N u(x_j) = \sum_{k=0}^{N} \tilde{u}_k p_k(x_j). \tag{1.2.24}$$

One often needs to determine $\{\tilde{u}_k\}$ from $\{u(x_j)\}$ or vice versa. A naive approach is to consider (1.2.24) as a linear system with unknowns $\{\tilde{u}_k\}$ and use a direct method, such as Gaussian elimination, to determine $\{\tilde{u}_k\}$. This approach requires $\mathcal{O}(N^3)$ operations and is not only too expensive but also often unstable due to roundoff errors. We shall now describe a stable $\mathcal{O}(N^2)$-approach using the properties of orthogonal polynomials.

A direct consequence of Gauss-quadrature is the following:

Lemma 1.2.4 *Let $x_0, x_1, \cdots, x_N$ be the zeros of the orthogonal polynomial p_{N+1}, and let $\{\omega_j\}$ be the associated Gauss-quadrature weights. Then*

$$\sum_{n=0}^{N} p_i(x_n)p_j(x_n)\omega_n = 0, \qquad \textit{if} \quad i \neq j \leqslant N. \tag{1.2.25}$$

We derive from (1.2.24) and (1.2.25) that

$$\sum_{j=0}^{N} u(x_j)p_l(x_j)\omega_j = \sum_{j=0}^{N}\sum_{k=0}^{N} \tilde{u}_k p_k(x_j)p_l(x_j)\omega_j = \tilde{u}_l(p_l, p_l)_{N,\omega}. \tag{1.2.26}$$

Hence, assuming the values of $\{p_j(x_k)\}$ are precomputed and stored as an $(N+1)\times(N+1)$ matrix, the forward transform (1.2.24) and the backward transform (1.2.26) can be performed by a simple matrix-vector multiplication which costs $\mathcal{O}(N^2)$ operations. We shall see in later sections that the $\mathcal{O}(N^2)$ operations can be improved to $\mathcal{O}(N\log N)$ if special orthogonal polynomials are used.

Exercise 1.2

Problem 1 Let $\omega(x) \equiv 1$ and $(a, b) = (-1, 1)$. Derive the three-term recurrence relation and compute the zeros of the corresponding orthogonal polynomial $P_7(x)$.

Problem 2 Prove Lemma 1.2.2.

Problem 3 Prove Lemma 1.2.4.

1.3 Chebyshev and Legendre polynomials

Chebyshev polynomials
Discrete norm and discrete Chebyshev transform
Legendre polynomials
Zeros of the Legendre polynomials
Discrete norm and discrete Legendre transform

The two most commonly used sets of orthogonal polynomials are the Chebyshev and Legendre polynomials. In this section, we will collect some of their basic properties.

Chebyshev polynomials

The Chebyshev polynomials $\{T_n(x)\}$ are generated from (1.2.4) with $\omega(x) = (1-x^2)^{-\frac{1}{2}}$, $(a, b) = (-1, 1)$ and normalized with $T_n(1) = 1$. They satisfy the

following three-term recurrence relation

$$\begin{aligned} &T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \qquad n \geqslant 1, \\ &T_0(x) \equiv 1, \quad T_1(x) = x, \end{aligned} \tag{1.3.1}$$

and the orthogonality relation

$$\int_{-1}^{1} T_k(x)T_j(x)(1-x^2)^{-\frac{1}{2}}\mathrm{d}x = \frac{c_k\pi}{2}\delta_{kj}, \tag{1.3.2}$$

where $c_0 = 2$ and $c_k = 1$ for $k \geqslant 1$. A unique feature of the Chebyshev polynomials is their explicit relation with a trigonometric function:

$$T_n(x) = \cos\left(n\cos^{-1}x\right), \quad n = 0, 1, \cdots. \tag{1.3.3}$$

One may derive from the above many special properties, e.g., it follows from (1.3.3) that

$$\begin{aligned} &2T_n(x) = \frac{1}{n+1}T'_{n+1}(x) - \frac{1}{n-1}T'_{n-1}(x), \qquad n \geqslant 2, \\ &T_0(x) = T'_1(x), \quad 2T_1(x) = \frac{1}{2}T'_2(x). \end{aligned} \tag{1.3.4}$$

One can also infer from (1.3.3) that $T_n(x)$ has the same parity as n. Moreover, we can derive from (1.3.4) that

$$T'_n(x) = 2n \sum_{\substack{k=0 \\ k+n \text{ odd}}}^{n-1} \frac{1}{c_k}T_k(x), \qquad T''_n(x) = \sum_{\substack{k=0 \\ k+n \text{ even}}}^{n-2} \frac{1}{c_k}n(n^2-k^2)T_k(x). \tag{1.3.5}$$

By (1.3.3), it can be easily shown that

$$|T_n(x)| \leqslant 1, \qquad |T'_n(x)| \leqslant n^2, \tag{1.3.6a}$$

$$T_n(\pm 1) = (\pm 1)^n, \qquad T'_n(\pm 1) = (\pm 1)^{n-1}n^2, \tag{1.3.6b}$$

$$2T_m(x)T_n(x) = T_{m+n}(x) + T_{m-n}(x), \qquad m \geqslant n. \tag{1.3.6c}$$

The Chebyshev polynomials $\{T_k(x)\}$ can also be defined as the normalized eigenfunctions of the singular Sturm-Liouville problem

$$\left(\sqrt{1-x^2}T'_k(x)\right)' + \frac{k^2}{\sqrt{1-x^2}}T_k(x) = 0, \qquad x \in (-1, 1). \tag{1.3.7}$$

We infer from the above and (1.3.2) that

$$\int_{-1}^{1} T_k'(x)T_j'(x)\sqrt{1-x^2}\mathrm{d}x = \frac{c_k k^2 \pi}{2}\delta_{kj}, \tag{1.3.8}$$

i.e. the polynomials $\{T_k'(x)\}$ are mutually orthogonal with respect to the weight function $w(x) = \sqrt{1-x^2}$.

An important feature of the Chebyshev polynomials is that the Gauss-type quadrature points and weights can be expressed explicitly as follows.①

Chebyshev-Gauss:

$$x_j = \cos\frac{(2j+1)\pi}{2N+2}, \quad \omega_j = \frac{\pi}{N+1}, \quad 0 \leqslant j \leqslant N. \tag{1.3.9}$$

Chebyshev-Gauss-Radau:

$$x_0 = 1, \quad \omega_0 = \frac{\pi}{2N+1}, \quad x_j = \cos\frac{2\pi j}{2N+1}, \quad \omega_j = \frac{2\pi}{2N+1}, \quad 1 \leqslant j \leqslant N. \tag{1.3.10}$$

Chebyshev-Gauss-Lobatto:

$$x_0 = 1, \quad x_N = -1, \quad \omega_0 = \omega_N = \frac{\pi}{2N}, \quad x_j = \cos\frac{\pi j}{N}, \quad \omega_j = \frac{\pi}{N}, \quad 1 \leqslant j \leqslant N-1. \tag{1.3.11}$$

Discrete norm and discrete Chebyshev transform

For the discrete norm $\|\cdot\|_{N,\omega}$ associated with the Gauss or Gauss-Radau quadrature, we have $\|u\|_{N,\omega} = \|u\|_\omega$ for all $u \in P_N$. For the discrete norm $\|\cdot\|_{N,\omega}$ associated with the Chebyshev-Gauss-Lobatto quadrature, the following result holds.

Lemma 1.3.1 *For all $u \in P_N$,*

$$\|u\|_{L^2_\omega} \leqslant \|u\|_{N,\omega} \leqslant \sqrt{2}\|u\|_{L^2_\omega}. \tag{1.3.12}$$

Proof For $u = \sum_{k=0}^{N} \tilde{u}_k T_k$, we have

$$\|u\|^2_{L^2_\omega} = \sum_{k=0}^{N} \tilde{u}_k^2 \frac{c_k \pi}{2}. \tag{1.3.13}$$

On the other hand,

① For historical reasons and for simplicity of notation, the Chebyshev points are often ordered in descending order. We shall keep this convention in this book.

$$\|u\|_{N,\omega}^2 = \sum_{k=0}^{N-1} \tilde{u}_k^2 \frac{c_k \pi}{2} + \tilde{u}_N^2 \langle T_N, T_N \rangle_{N,\omega}. \tag{1.3.14}$$

The inequality (1.3.12) follows from the above results and the identity

$$(T_N, T_N)_{N,\omega} = \sum_{j=0}^{N} \frac{\pi}{\tilde{c}_j N} \cos^2 j\pi = \pi, \tag{1.3.15}$$

where $\tilde{c}_0 = \tilde{c}_N = 2$ and $\tilde{c}_k = 1$ for $1 \leqslant k \leqslant N-1$. □

Let $\{\xi_i\}_{i=0}^N$ be the Chebyshev-Gauss-Lobatto points, i.e. $\xi_i = \cos(i\pi/N)$, and let u be a continuous function on $[-1, 1]$. We write

$$u(\xi_i) = I_N u(\xi_i) = \sum_{k=0}^{N} \tilde{u}_k T_k(\xi_i) = \sum_{k=0}^{N} \tilde{u}_k \cos(ki\pi/N), \quad i = 0, 1, \cdots, N. \tag{1.3.16}$$

One derives immediately from the Chebyshev-Gauss-quadrature that

$$\tilde{u}_k = \frac{2}{\tilde{c}_k N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} u(\xi_j) \cos(kj\pi/N). \tag{1.3.17}$$

The main advantage of using Chebyshev polynomials is that the backward and forward discrete Chebyshev transforms (1.3.16) and (1.3.17) can be performed in $\mathcal{O}(N \log_2 N)$ operations, thanks to the Fast Fourier Transform (FFT), see Section 1.5. The main disadvantage is that the Chebyshev polynomials are mutually orthogonal with respect to a singular weight function $(1-x^2)^{-\frac{1}{2}}$ which introduces significant difficulties in the analysis of the Chebyshev spectral method.

Legendre polynomials

The Legendre polynomials $\{L_n(x)\}$ are generated from (1.2.4) with $\omega(x) \equiv 1$, $(a, b) = (-1, 1)$ and the normalization $L_n(1) = 1$. The Legendre polynomials satisfy the three-term recurrence relation

$$\begin{aligned} &L_0(x) = 1, \qquad L_1(x) = x, \\ &(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x), \quad n \geqslant 1, \end{aligned} \tag{1.3.18}$$

and the orthogonality relation

$$\int_{-1}^{1} L_k(x) L_j(x) \mathrm{d}x = \frac{1}{k + \frac{1}{2}} \delta_{kj}. \tag{1.3.19}$$

The Legendre polynomials can also be defined as the normalized eigenfunctions of the singular Sturm-Liouville problem

$$\big((1-x^2)L_n'(x)\big)' + n(n+1)L_n(x) = 0, \qquad x \in (-1,1), \tag{1.3.20}$$

from which and (1.3.19) we infer that

$$\int_{-1}^{1} L_k'(x)L_j'(x)(1-x^2)\mathrm{d}x = \frac{k(k+1)}{k+\frac{1}{2}}\delta_{kj}, \tag{1.3.21}$$

i.e. the polynomials $\{L_k'(x)\}$ are mutually orthogonal with respect to the weight function $\omega(x) = 1-x^2$.

Other useful properties of the Legendre polynomials include:

$$\int_{-1}^{x} L_n(\xi)\mathrm{d}\xi = \frac{1}{2n+1}(L_{n+1}(x) - L_{n-1}(x)), \qquad n \geqslant 1; \tag{1.3.22a}$$

$$L_n(x) = \frac{1}{2n+1}(L_{n+1}'(x) - L_{n-1}'(x)); \tag{1.3.22b}$$

$$L_n(\pm 1) = (\pm 1)^n, \qquad L_n'(\pm 1) = \frac{1}{2}(\pm 1)^{n-1}n(n+1); \tag{1.3.22c}$$

$$L_n'(x) = \sum_{\substack{k=0 \\ k+n \text{ odd}}}^{n-1} (2k+1)L_k(x); \tag{1.3.22d}$$

$$L_n''(x) = \sum_{\substack{k=0 \\ k+n \text{ even}}}^{n-2} \left(k+\frac{1}{2}\right)\left(n(n+1) - k(k+1)\right)L_k(x). \tag{1.3.22e}$$

For the Legendre series, the quadrature points and weights are

Legendre-Gauss: x_j are the zeros of $L_{N+1}(x)$, and

$$\omega_j = \frac{2}{(1-x_j^2)[L_{N+1}'(x_j)]^2}, \qquad 0 \leqslant j \leqslant N. \tag{1.3.23}$$

Legendre-Gauss-Radau: x_j are the $N+1$ zeros of $L_N(x) + L_{N+1}(x)$, and

$$\omega_0 = \frac{2}{(N+1)^2}, \qquad \omega_j = \frac{1}{(N+1)^2}\,\frac{1-x_j}{[L_N(x_j)]^2}, \quad 1 \leqslant j \leqslant N. \tag{1.3.24}$$

Legendre-Gauss-Lobatto: $x_0 = -1, x_N = 1$, $\{x_j\}_{j=1}^{N-1}$ are the zeros of $L_N'(x)$, and

$$\omega_j = \frac{2}{N(N+1)}\,\frac{1}{[L_N(x_j)]^2}, \qquad 0 \leqslant j \leqslant N. \tag{1.3.25}$$

Zeros of Legendre polynomials

We observe from the last subsection that the three types of quadrature points for the Legendre polynomials are related to the zeros of the L_{N+1}, $L_{N+1}+L_N$ and L'_N.

Theorem 1.2.1 provides a simple and efficient way to compute the zeros of orthogonal polynomials, given the three-term recurrence relation. However, this method may suffer from round-off errors as N becomes very large. As a result, we will present an alternative method to compute the zeros of $L_N^{(m)}(x)$ numerically, where $m < N$ is the order of derivative.

We start from the left boundary -1 and try to find the small interval of width H which contains the first zero z_1. The idea for locating the interval is similar to that used by the *bisection method*. In the resulting (small) interval, we use *Newton's method* to find the first zero. The Newton's method for finding a root of $f(x) = 0$ is

$$x_{k+1} = x_k - f(x_k)/f'(x_k). \tag{1.3.26}$$

After finding the first zero, we use the point $z_1 + H$ as the starting point and repeat the previous procedure to get the second zero z_2. This will give us all the zeros of $L_N^{(m)}(x)$. The parameter H, which is related to the smallest gap of the zeros, will be chosen as N^{-2}.

The following pseudo-code generates the zeros of $L_N^{(m)}(x)$.

```
CODE  LGauss.1
Input N, ϵ, m  %ϵ is the accuracy tolerence
H=N^-2; a=-1
For k=1 to N-m do
%The following is to search the small interval containing
a root
   b=a+H
   while L_N^(m)(a)*L_N^(m)(b) > 0
     a=b; b=a+H
   endwhile
%the Newton's method in (a,b)
        x=(a+b)/2;  xright=b
        while |x-xright|⩾ϵ
          xright=x; x=x-L_N^(m)(x)/L_N^(m+1)(x)
        endwhile
        z(k)=x
   a=x+H %move to another interval containing a root
endFor
Output z(1), z(2),···,z(N-m)
```

In the above pseudo-code, the parameter ϵ is used to control the accuracy of the zeros. Also, we need to use the recurrence formulas (1.3.18) and (1.3.22b) to obtain $L_n^{(m)}(x)$ which are used in the above code.

```
CODE LGauss.2
%This code is to evaluate L_n^(m)(x).
function r=Legendre(n,m,x)
For j=0 to m do
   If j=0 then
      s(0,j)=1;  s(1,j)=x
      for k=1 to n-1 do
        s(k+1,j)=((2k+1)*x*s(k,j)-k*s(k-1,j))/(k+1)
      endfor
   else  s(0,j)=0
      if j=1 then  s(1,j)=2
        else s(1,j)=0
      endif
      for k=1 to n-1 do
        s(k+1,j)=(2k+1)*s(k,j-1)+s(k-1,j)
      endfor
   endIf
endFor
r=s(n,m)
```

As an example, by setting $N = 7, m = 0$ and $\epsilon = 10^{-8}$ in `CODE LGauss.1`, we obtain the zeros for $L_7(x)$:

z_1	-0.94910791	z_5	0.40584515
z_2	-0.74153119	z_6	0.74153119
z_3	-0.40584515	z_7	0.94910791
z_4	0.00000000		

By setting $N = 6, m = 1$ and $\epsilon = 10^{-8}$ in `CODE LGauss.1`, we obtain the zeros for $L_6'(x)$. Together with $Z_1 = -1$ and $Z_7 = 1$, they form the Legendre-Gauss-Lobatto points:

Z_1	-1.00000000	Z_5	0.46884879
Z_2	-0.83022390	Z_6	0.83022390
Z_3	-0.46884879	Z_7	1.00000000
Z_4	0.00000000		

Discrete norm and discrete Legendre transform

As opposed to the Chebyshev polynomials, the main advantage of Legendre poly-

nomialsis that they are mutually orthogonal in the standard L^2-inner product, so the analysis of Legendre spectral methods is much easier than that of the Chebyshev spectral method. The main disadvantage is that there is no practical fast discrete Legendre transform available. However, it is possible to take advantage of both the Chebyshev and Legendre polynomials by constructing the so called Chebyshev-Legendre spectral methods; we refer to [41] and [141] for more details.

Lemma 1.3.2 *Let $\|\cdot\|_N$ be the discrete norm relative to the Legendre-Gauss-Lobatto quadrature. Then*

$$\|u\|_{L^2} \leqslant \|u\|_N \leqslant \sqrt{3}\|u\|_{L^2}, \quad \textit{for all} \quad u \in P_N. \tag{1.3.27}$$

Proof Setting $u = \sum_{k=0}^{N} \tilde{u}_k L_k$, we have from (1.3.19) that $\|u\|_{L^2}^2 = \sum_{k=0}^{N} 2\tilde{u}_k^2/(2k+1)$. On the other hand,

$$\|u\|_N^2 = \sum_{k=0}^{N-1} \tilde{u}_k^2 \frac{2}{2k+1} + \tilde{u}_N^2 (L_N, L_N)_N.$$

The desired result (1.3.27) follows from the above results, the identity

$$(L_N, L_N)_N = \sum_{j=0}^{N} L_N(x_j)^2 \omega_j = 2/N, \tag{1.3.28}$$

and the fact that $\frac{2}{2N+1} \leqslant \frac{2}{N} \leqslant 3\frac{2}{2N+1}$. □

Let $\{x_i\}_{0 \leqslant i \leqslant N}$ be the Legendre-Gauss-Lobatto points, and let u be a continuous function on $[-1, 1]$. We may write

$$u(x_j) = I_N u(x_j) = \sum_{k=0}^{N} \tilde{u}_k L_k(x_j). \tag{1.3.29}$$

We then derive from the Legendre-Gauss-Lobatto quadrature points that the discrete Legendre coefficients $\tilde{u}_k$ can be determined by the relation

$$\tilde{u}_k = \frac{1}{N+1} \sum_{j=0}^{N} u(x_j) \frac{L_k(x_j)}{L_N(x_j)}, \qquad k = 0, 1, \cdots, N. \tag{1.3.30}$$

The values $\{L_k(x_j)\}$ can be pre-computed and stored as a $(N+1)\times(N+1)$ matrix by using the three-term recurrence relation (1.3.18). Hence, the backward and forward

discrete Legendre transforms (1.3.30) and (1.3.29) can be performed by a matrix-vector multiplication which costs $\mathcal{O}(N^2)$ operations.

Exercise 1.3

Problem 1 Prove (1.3.22).

Problem 2 Derive the three-term recurrence relation for $\{L_k + L_{k+1}\}$ and use the method in Theorem 1.2.1 to find the Legendre-Gauss-Radau points with $N = 16$.

Problem 3 Prove (1.3.30).

1.4 Jacobi polynomials and generalized Jacobi polynomials

Basic properties of Jacobi polynomials
Generalized Jacobi polynomials

An important class of orthogonal polynomials are the so called Jacobi polynomials, which are denoted by $J_n^{\alpha,\beta}(x)$ and generated from (1.2.4) with

$$\omega(x) = (1-x)^\alpha (1+x)^\beta \quad \text{for } \alpha,\ \beta > -1,\ (a,b) = (-1,1), \tag{1.4.1}$$

and normalized by

$$J_n^{\alpha,\beta}(1) = \frac{\Gamma(n+\alpha+1)}{n!\Gamma(\alpha+1)}, \tag{1.4.2}$$

where $\Gamma(x)$ is the usual Gamma function. In fact, both the Chebyshev and Legendre polynomials are special cases of the Jacobi polynomials, namely, the Chebyshev polynomials $T_n(x)$ correspond to $\alpha = \beta = -\frac{1}{2}$ with the normalization $T_n(1) = 1$, and the Legendre polynomials $L_n(x)$ correspond to $\alpha = \beta = 0$ with the normalization $L_n(1) = 1$.

Basic properties of Jacobi polynomials

We now present some basic properties of the Jacobi polynomials which will be frequently used in the implementation and analysis of spectral methods. We refer to [155] for a complete and authoritative presentation of the Jacobi polynomials.

The three-term recurrence relation for the Jacobi polynomials is:

$$\begin{aligned} &J_{n+1}^{\alpha,\beta}(x) = (a_n^{\alpha,\beta} x - b_n^{\alpha,\beta}) J_n^{\alpha,\beta}(x) - c_n^{\alpha,\beta} J_{n-1}^{\alpha,\beta}(x), \quad n \geqslant 1, \\ &J_0^{\alpha,\beta}(x) = 1, \quad J_1^{\alpha,\beta}(x) = \frac{1}{2}(\alpha+\beta+2)x + \frac{1}{2}(\alpha-\beta), \end{aligned} \tag{1.4.3}$$

where

$$a_n^{\alpha,\beta} = \frac{(2n+\alpha+\beta+1)(2n+\alpha+\beta+2)}{2(n+1)(n+\alpha+\beta+1)}, \tag{1.4.4a}$$

$$b_n^{\alpha,\beta} = \frac{(\beta^2-\alpha^2)(2n+\alpha+\beta+1)}{2(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)}, \tag{1.4.4b}$$

$$c_n^{\alpha,\beta} = \frac{(n+\alpha)(n+\beta)(2n+\alpha+\beta+2)}{(n+1)(n+\alpha+\beta+1)(2n+\alpha+\beta)}. \tag{1.4.4c}$$

The Jacobi polynomials satisfy the orthogonality relation

$$\int_{-1}^{1} J_n^{\alpha,\beta}(x) J_m^{\alpha,\beta}(x)(1-x)^{\alpha}(1+x)^{\beta}\mathrm{d}x = 0 \quad \text{for } n \neq m. \tag{1.4.5}$$

A property of fundamental importance is the following:

Theorem 1.4.1 *The Jacobi polynomials satisfy the following singular Sturm-Liouville problem*:

$$(1-x)^{-\alpha}(1+x)^{-\beta}\frac{d}{dx}\left\{(1-x)^{\alpha+1}(1+x)^{\beta+1}\frac{d}{dx}J_n^{\alpha,\beta}(x)\right\} + n(n+1+\alpha+\beta)J_n^{\alpha,\beta}(x) = 0, \quad -1 < x < 1.$$

Proof We denote $\omega(x) = (1-x)^{\alpha}(1+x)^{\beta}$. By applying integration by parts twice, we find that for any $\phi \in P_{n-1}$,

$$\int_{-1}^{1} \frac{\mathrm{d}}{\mathrm{d}x}\left\{(1-x)^{\alpha+1}(1+x)^{\beta+1}\frac{dJ_n^{\alpha,\beta}}{dx}\right\}\phi dx = -\int_{-1}^{1}\omega(1-x^2)\frac{\mathrm{d}J_n^{\alpha,\beta}}{dx}\frac{\mathrm{d}\phi}{\mathrm{d}x}\mathrm{d}x$$

$$= \int_{-1}^{1} J_n^{\alpha,\beta}\left\{[-(\alpha+1)(1+x)+(\beta+1)(1-x)]\frac{\mathrm{d}\phi}{\mathrm{d}x} + (1-x^2)\frac{\mathrm{d}^2\phi}{\mathrm{d}x^2}\right\}\omega\mathrm{d}x = 0.$$

The last equality follows from the fact that $\int_{-1}^{1} J_n^{\alpha,\beta}\psi\omega(x)\mathrm{d}x = 0$ for any $\psi \in P_{n-1}$. An immediate consequence of the above relation is that there exists λ such that

$$-\frac{d}{dx}\left\{(1-x)^{\alpha+1}(1+x)^{\beta+1}\frac{d}{dx}J_n^{\alpha,\beta}(x)\right\} = \lambda J_n^{\alpha,\beta}(x)\omega(x).$$

To determine λ, we take the coefficients of the leading term $x^{n+\alpha+\beta}$ in the above relation. Assuming that $J_n^{\alpha,\beta}(x) = k_n x^n + \{\text{lower order terms}\}$, we get $k_n n(n+1+$

$\alpha+\beta)=k_n\lambda$, which implies that $\lambda=n(n+1+\alpha+\beta)$. □

From Theorem 1.4.1 and (1.4.5), one immediately derives the following result:

Lemma 1.4.1 *For $n\neq m$,*

$$\int_{-1}^{1}(1-x)^{\alpha+1}(1+x)^{\beta+1}\frac{\mathrm{d}J_n^{\alpha,\beta}}{\mathrm{d}x}\frac{\mathrm{d}J_m^{\alpha,\beta}}{\mathrm{d}x}\mathrm{d}x=0. \quad \square \tag{1.4.6}$$

The above relation indicates that $\frac{d}{dx}J_n^{\alpha,\beta}$ forms a sequence of orthogonal polynomials with weight $\omega(x)=(1-x)^{\alpha+1}(1+x)^{\beta+1}$. Hence, by the uniqueness, we find that $\frac{d}{dx}J_n^{\alpha,\beta}$ is proportional to $J_{n-1}^{\alpha+1,\beta+1}$. In fact, we can prove the following important derivative recurrence relation:

Lemma 1.4.2 *For $\alpha,\beta>-1$,*

$$\partial_x J_n^{\alpha,\beta}(x)=\frac{1}{2}(n+\alpha+\beta+1)J_{n-1}^{\alpha+1,\beta+1}(x). \quad \square \tag{1.4.7}$$

Generalized Jacobi polynomials

Since for $\alpha\leqslant -1$ and/or $\beta\leqslant -1$, the function $\omega^{\alpha,\beta}$ is not in $L^1(I)$ so it cannot be used as a usual weight function. Hence, the classical Jacobi polynomials are only defined for $\alpha,\beta>-1$. However, as we shall see later, it is very useful to extend the definition of $J_n^{\alpha,\beta}$ to the cases where α and/or β are negative integers.

We now define the *generalized Jacobi polynomials* (GJPs) with integer indexes (k,l). Let us denote

$$n_0:=n_0(k,l)=\begin{cases}-(k+l) & \text{if } k,l\leqslant -1,\\ -k & \text{if } k\leqslant -1,\ l>-1,\\ -l & \text{if } k>-1,\ l\leqslant -1,\end{cases} \tag{1.4.8}$$

Then, the GJPs are defined as

$$J_n^{k,l}(x)=\begin{cases}(1-x)^{-k}(1+x)^{-l}J_{n-n_0}^{-k,-l}(x) & \text{if } k,l\leqslant -1,\\ (1-x)^{-k}J_{n-n_0}^{-k,l}(x) & \text{if } k\leqslant -1,\ l>-1,\\ (1+x)^{-l}J_{n-n_0}^{k,-l}(x) & \text{if } k>-1,\ l\leqslant -1,\end{cases}\qquad n\geqslant n_0. \tag{1.4.9}$$

It is easy to verify that $J_n^{k,l}\in P_n$.

We now present some important properties of the GJPs. First of all, it is easy to check that the GJPs are orthogonal with the generalized Jacobi weight $\omega^{k,l}$ for all

integers k and l, i.e.,

$$\int_{-1}^{1} J_n^{k,l}(x) J_m^{k,l}(x) \omega^{k,l}(x) \mathrm{d}x = 0, \quad \forall n \neq m. \tag{1.4.10}$$

It can be shown that the GJPs with negative integer indexes can be expressed as compact combinations of Legendre polynomials.

Lemma 1.4.3 *Let $k, l \geqslant 1$ and $k, l \in \mathbb{Z}$. There exists a set of constants $\{a_j\}$ such that*

$$J_n^{-k,-l}(x) = \sum_{j=n-k-l}^{n} a_j L_j(x), \quad n \geqslant k + l. \tag{1.4.11}$$

As some important special cases, one can verify that

$$\begin{aligned}
J_n^{-1,-1} &= \frac{2(n-1)}{2n-1}\Big(L_{n-2} - L_n\Big),\\
J_n^{-2,-1} &= \frac{2(n-2)}{2n-3}\Big(L_{n-3} - \frac{2n-3}{2n-1}L_{n-2} - L_{n-1} + \frac{2n-3}{2n-1}L_n\Big),\\
J_n^{-1,-2} &= \frac{2(n-2)}{2n-3}\Big(L_{n-3} + \frac{2n-3}{2n-1}L_{n-2} - L_{n-1} - \frac{2n-3}{2n-1}L_n\Big),\\
J_n^{-2,-2} &= \frac{4(n-1)(n-2)}{(2n-3)(2n-5)}\Big(L_{n-4} - \frac{2(2n-3)}{2n-1}L_{n-2} + \frac{2n-5}{2n-1}L_n\Big).
\end{aligned} \tag{1.4.12}$$

It can be shown (cf. [75]) that the generalized Jacobi polynomials satisfy the derivative recurrence relation stated in the following lemma.

Lemma 1.4.4 *For $k, l \in \mathbb{Z}$, we have*

$$\partial_x J_n^{k,l}(x) = C_n^{k,l} J_{n-1}^{k,l}(x), \tag{1.4.13}$$

where

$$C_n^{k,l} = \begin{cases} -2(n+k+l+1) & \text{if} \quad k, l \leqslant -1,\\ -n & \text{if} \quad k \leqslant -1,\ l > -1,\\ -n & \text{if} \quad k > -1,\ l \leqslant -1,\\ \dfrac{1}{2}(n+k+l+1) & \text{if} \quad k, l > -1. \end{cases} \tag{1.4.14}$$

Remark 1.4.1 Since $\omega^{\alpha,\beta} \notin L^1(I)$ for $\alpha \leqslant -1$ and $\beta \leqslant -1$, it is necessary that the generalized Jacobi polynomials vanish at one or both end points. In fact, an important feature of the GJPs is that for $k, l \geqslant 1$, we have

$$\begin{aligned} &\partial_x^i J_n^{-k,-l}(1) = 0, \qquad i = 0, 1, \cdots, k-1; \\ &\partial_x^j J_n^{-k,-l}(-1) = 0, \quad j = 0, 1, \cdots, l-1. \end{aligned} \tag{1.4.15}$$

Thus, they can be directly used as basis functions for boundary-value problems with corresponding boundary conditions.

Exercise 1.4

Problem 1 Prove (1.4.12) by the definition (1.4.9).

Problem 2 Prove Lemma 1.4.4.

1.5 Fast Fourier transform

Two basic lemmas
Computational cost
Tree diagram
Fast inverse Fourier transform
Fast Cosine transform
The discrete Fourier transform

Much of this section will be using complex exponentials. We first recall *Euler's formula*: $e^{i\theta} = \cos\theta + i\sin\theta$, where $i = \sqrt{-1}$. It is also known that the functions E_k defined by

$$E_k(x) = e^{ikx}, \qquad k = 0, \pm 1, \cdots \tag{1.5.1}$$

form an orthogonal system of functions in the complex space $L_2[0, 2\pi]$, provided that we define the inner-product to be

$$\langle f, g\rangle = \frac{1}{2\pi}\int_0^{2\pi} f(x)\overline{g(x)}\mathrm{d}x.$$

This means that $\langle E_k, E_m\rangle = 0$ when $k \neq m$, and $\langle E_k, E_k\rangle = 1$. For discrete values, it will be convenient to use the following inner-product notation:

$$\langle f, g\rangle_N = \frac{1}{N}\sum_{j=0}^{N-1} f\left(x_j\right)\overline{g\left(x_j\right)}, \tag{1.5.2}$$

where

$$x_j = 2\pi j/N, \quad 0 \leqslant j \leqslant N-1. \tag{1.5.3}$$

The above is not a true inner-product because the condition $\langle f, f\rangle_N = 0$ does not

imply $f \equiv 0$. It implies that $f(x)$ takes the value 0 at each node x_j.

The following property is important.

Lemma 1.5.1 *For any $N \geqslant 1$, we have*

$$\langle E_k, E_m \rangle_N = \begin{cases} 1 & \text{if } k-m \text{ is divisible by } N, \\ 0 & \text{otherwise.} \end{cases} \tag{1.5.4}$$

A 2π-periodic function $p(x)$ is said to be an *exponential polynomial* of degree at most n if it can be written in the form

$$p(x) = \sum_{k=0}^{n} c_k e^{ikx} = \sum_{k=0}^{n} c_k E_k(x). \tag{1.5.5}$$

The coefficients $\{c_k\}$ can be determined by taking the discrete inner-product of (1.5.5) with E_m. More precisely, it follows from (1.5.4) that the coefficients $c_0, c_1, \cdots, c_{N-1}$ in (1.5.5) can be expressed as:

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f(x_j) e^{-ikx_j}, \qquad 0 \leqslant k \leqslant N-1, \tag{1.5.6}$$

where x_j is defined by (1.5.3). In practice, one often needs to determine $\{c_k\}$ from $\{f(x_j)\}$, or vice versa. It is clear that a direct computation using (1.5.6) requires $\mathcal{O}(N^2)$ operations. In 1965, a paper by Cooley and Tukey [33] described a different method of calculating the coefficients c_k, $0 \leqslant k \leqslant N-1$. The method requires only $\mathcal{O}(N \log_2 N)$ multiplications and $\mathcal{O}(N \log_2 N)$ additions, provided N is chosen in an appropriate manner. For a problem with thousands of data points, this reduces the number of calculations to thousands compared to millions for the direct technique.

The method described by Cooley and Tukey has become to be known either as the Cooley-Tukey Algorithm or the Fast Fourier Transform (FFT) Algorithm, and has led to a revolution in the use of interpolating trigonometric polynomials. We follow the exposition of Kincaid and Cheney [92] to introduce the algorithm.

Two basic lemmas

Lemma 1.5.2 *Let p and q be exponential polynomials of degree $N-1$ such that, for the points $y_j = \pi j/N$, we have*

$$p(y_{2j}) = f(y_{2j}), \qquad q(y_{2j}) = f(y_{2j+1}), \qquad 0 \leqslant j \leqslant N-1. \tag{1.5.7}$$

Then the exponential polynomial of degree $\leqslant 2N-1$ *that interpolates* f *at the points* $y_j, 0 \leqslant j \leqslant 2N-1$, *is given by*

$$P(x) = \frac{1}{2}(1+e^{iNx})p(x) + \frac{1}{2}(1-e^{iNx})q(x-\pi/N). \tag{1.5.8}$$

Proof Since p and q have degrees $\leqslant N-1$, whereas e^{jNx} is of degree N, it is clear that P has degree $\leqslant 2N-1$. It remains to show that P interpolates f at the nodes. We have, for $0 \leqslant j \leqslant 2N-1$,

$$P(y_j) = \frac{1}{2}(1+E_N(y_j))p(y_j) + \frac{1}{2}(1-E_N(y_j))q(y_j-\pi/N).$$

Notice that $E_N(y_j) = (-1)^j$. Thus for even j, we infer that $P(y_j) = p(y_j) = f(y_j)$, whereas for odd j, we have

$$P(y_j) = q(y_j - \pi/N) = q(y_{j-1}) = f(y_j).$$

This completes the proof of Lemma 1.5.2. □

Lemma 1.5.3 *Let the coefficients of the polynomials described in Lemma 1.5.2 be as follows*:

$$p = \sum_{j=0}^{N-1} \alpha_j E_j, \qquad q = \sum_{j=0}^{N-1} \beta_j E_j, \qquad P = \sum_{j=0}^{2N-1} \gamma_j E_j.$$

Then, for $0 \leqslant j \leqslant N-1$,

$$\gamma_j = \frac{1}{2}\alpha_j + \frac{1}{2}e^{-ij\pi/N}\beta_j, \quad \gamma_{j+N} = \frac{1}{2}\alpha_j - \frac{1}{2}e^{-ij\pi/N}\beta_j. \tag{1.5.9}$$

Proof To prove (1.5.9), we will be using (1.5.8) and will require a formula for $q(x-\pi/N)$:

$$q(x-\pi/N) = \sum_{j=0}^{N-1} \beta_j E_j(x-\pi/N) = \sum_{j=0}^{N-1} \beta_j e^{ij(x-\pi/N)} = \sum_{j=0}^{N-1} \beta_j e^{-i\pi j/N} E_j(x).$$

Thus, from equation (1.5.8),

$$P = \frac{1}{2}\sum_{j=0}^{N-1} \left\{(1+E_N)\alpha_j E_j + (1-E_N)\beta_j e^{-i\pi j/N} E_j\right\}$$

$$= \frac{1}{2} \sum_{j=0}^{N-1} \Big\{ (\alpha_j + \beta_j e^{-ij\pi/N}) E_j + (\alpha_j - \beta_j e^{-ij\pi/N}) E_{N+j} \Big\}.$$

The formulas for the coefficients γ_j can now be read from this equation. This completes the proof of Lemma 1.5.3. □

Computational cost

It follows from (1.5.6), (1.5.7) and (1.5.8) that

$$\alpha_j = \frac{1}{N} \sum_{j=0}^{N-1} f(x_{2j}) e^{-2\pi ij/N},$$

$$\beta_j = \frac{1}{N} \sum_{j=0}^{N-1} f(x_{2j+1}) e^{-2\pi ij/N},$$

$$\gamma_j = \frac{1}{2N} \sum_{j=0}^{2N-1} f(x_j) e^{-\pi ij/N}.$$

For the further analysis, let $R(N)$ denote the minimum number of multiplications necessary to compute the coefficients in an interpolating exponential polynomial for the set of points $\{2\pi j/N : 0 \leqslant j \leqslant N-1\}$.

First, we can show that

$$R(2N) \leqslant 2R(N) + 2N. \tag{1.5.10}$$

It is seen that $R(2N)$ is the minimum number of multiplications necessary to compute γ_j, and $R(N)$ is the minimum number of multiplications necessary to compute α_j or β_j. By Lemma 1.5.3, the coefficients γ_j can be obtained from α_j and β_j at the cost of $2N$ multiplications. Indeed, we require N multiplications to compute $\frac{1}{2}\alpha_j$ for $0 \leqslant j \leqslant N-1$, and another N multiplications to compute $(\frac{1}{2}e^{-ij\pi/N})\beta_j$ for $0 \leqslant j \leqslant N-1$. (In the latter, we assume that the factors $\frac{1}{2}e^{-ij\pi/N}$ have already been made available.) Since the cost of computing coefficients $\{\alpha_j\}$ is $R(N)$ multiplications, and the same is true for computing $\{\beta_j\}$, the total cost for P is at most $2R(N) + 2N$ multiplications. It follows from (1.5.10) and mathematical induction that $R(2^n) \leqslant m\, 2^m$. As a consequence of the above result, we see that if N is a power of 2, say 2^m, then the cost of computing the interpolating exponential polynomial obeys the inequality

$$R(N) \leqslant N \log_2 N.$$

The algorithm that carries out repeatedly the procedure in Lemma 1.5.2 is the fast Fourier transform.

Tree diagram

The content of Lemma 1.5.2 can be interpreted in terms of two linear operators, L_N and T_h. For any f, let $L_N f$ denote the exponential polynomial of degree $N-1$ that interpolates f at the nodes $2\pi j/N$ for $0 \leqslant j \leqslant N-1$. Let T_h be a *translation operator* defined by $(T_h f)(x) = f(x+h)$. We know from (1.5.4) that

$$L_N f = \sum_{k=0}^{N-1} < f,\ E_k >_N E_k.$$

Furthermore, in Lemma 1.5.2, $P = L_{2N} f, p = L_N f$ and $q = L_N T_{\pi/N} f$. The conclusion of Lemmas 1.5.2 and 1.5.3 is that $L_{2N} f$ can be obtained efficiently from $L_N f$ and $L_N T_{\pi/N} f$.

Our goal now is to establish one version of the fast Fourier transform algorithm for computing $L_N f$, where $N = 2^m$. We define

$$P_k^{(n)} = L_{2^n} T_{2k\pi/N} f, \qquad 0 \leqslant n \leqslant m,\ 0 \leqslant k \leqslant 2^{m-n} - 1. \tag{1.5.11}$$

An alternative description of $P_k^{(n)}$ is as the exponential polynomial of degree $2^n - 1$ that interpolates f in the following way:

$$P_k^{(n)}\left(\frac{2\pi j}{2^n}\right) = f\left(\frac{2\pi k}{N} + \frac{2\pi j}{2^n}\right), \qquad 0 \leqslant j \leqslant 2^n - 1.$$

A straightforward application of Lemma 1.5.2 shows that

$$P_k^{(n+1)}(x) = \frac{1}{2}\left(1 + e^{i2^n x}\right) P_k^{(n)} + \frac{1}{2}(1 - e^{i2^n x}) P_{k+2^{m-n-1}}^{(n)}\left(x - \frac{\pi}{2^n}\right). \tag{1.5.12}$$

We can illustrate in a *tree diagram* how the exponential polynomials $P_k^{(n)}$ are related. Suppose that our objective is to compute

$$P_0^{(3)} = L_8 f = \sum_{k=0}^{7} < f,\ E_k >_N E_k.$$

In accordance with (1.5.12), this function can be easily obtained from $P_0^{(2)}$ and $P_1^{(2)}$. Each of these, in turn, can be easily obtained from four polynomials of lower order,

and so on. Figure 1.2 shows the connections.

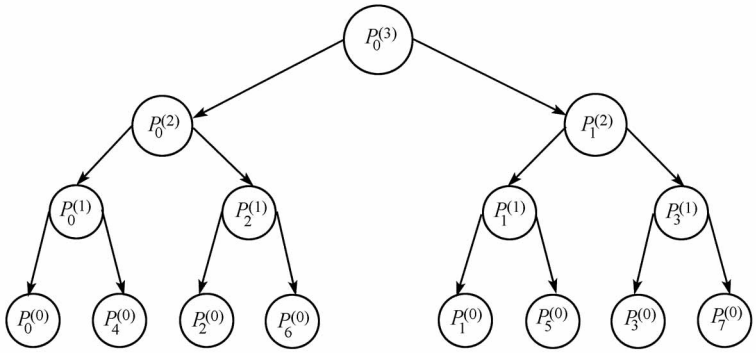

Figure 1.2 An illustration of a tree diagram

Algorithm

Denote the coefficients of $P_k^{(n)}$ by $A_{kj}^{(n)}$. Here $0 \leqslant n \leqslant m, 0 \leqslant k \leqslant 2^{m-n}-1$, and $0 \leqslant j \leqslant 2^n - 1$. We have

$$P_k^{(n)}(x) = \sum_{j=0}^{2^n-1} A_{kj}^{(n)} E_j(x) = \sum_{j=0}^{2^n-1} A_{kj}^{(n)} e^{ijx}.$$

By Lemma 1.5.3, the following equations hold:

$$A_{kj}^{(n+1)} = \frac{1}{2}\left[A_{kj}^{(n)} + e^{-ij\pi/2^n} A_{k+2^{m-n-1},\,j}^{(n)}\right],$$
$$A_{k,j+2^n}^{(n+1)} = \frac{1}{2}\left[A_{kj}^{(n)} - e^{-ij\pi/2^n} A_{k+2^{m-n-1},\,j}^{(n)}\right].$$

For a fixed n, the array $A^{(n)}$ requires $N = 2^m$ storage locations in memory because $0 \leqslant k \leqslant 2^{m-n}-1$ and $0 \leqslant j \leqslant 2^n - 1$. One way to carry out the computations is to use two linear arrays of length N, one to hold $A^{(n)}$ and the other to hold $A^{(n+1)}$. At the next stage, one array will contain $A^{(n+1)}$ and the other $A^{(n+2)}$. Let us call these arrays C and D. The two-dimensional array $A^{(n)}$ is stored in C by the rule

$$C(2^n k + j) = A_{kj}^{(n)}, \qquad 0 \leqslant k \leqslant 2^{m-n}-1, \quad 0 \leqslant j \leqslant 2^n - 1.$$

It is noted that if $0 \leqslant k, k' \leqslant 2^{m-n}-1$ and $0 \leqslant j, j' \leqslant 2^n - 1$ satisfying $2^n k + j =$

$2^n k' + j'$, then $(k, j) = (k', j')$. Similarly, the array $A^{(n+1)}$ is stored in D by the rule

$$D(2^{n+1}k + j) = A_{kj}^{(n+1)}, \qquad 0 \leqslant k \leqslant 2^{m-n-1} - 1, \quad 0 \leqslant j \leqslant 2^{n+1} - 1.$$

The factors $Z(j) = e^{-2\pi ij/N}$ are computed at the beginning and stored. Then we use the fact that

$$e^{-ij\pi/2^n} = Z(j2^{m-n-1}).$$

Below is the fast Fourier transform algorithm:

```
CODE FFT.1
% Cooley-Tukey Algorithm
Input  m
N=2^m, w=e^{-2πi/N}
for k=0 to N-1 do
    Z(k)=w^k, C(k)=f(2πk/N)
endfor
For n=0 to m-1 do
    for k=0 to 2^{m-n-1}-1 do
        for j=0 to 2^n-1 do
            u=C(2^n k+j); v=Z(j2^{m-n-1})*C(2^n k+2^{m-1}+j)
            D(2^{n+1}k+j)=0.5*(u+v); D(2^{n+1}k+j+2^n)=0.5*(u-v)
        endfor
    endfor
    for j=0 to N-1 do
        C(j)=D(j)
    endfor
endFor
Output C(0), C(1), ···, C(N-1).
```

By scrutinizing the pseudocode, we can also verify the bound $N \log_2 N$ for the number of multiplications involved. Notice that in the nested loop of the code, n takes on m values; then k takes on 2^{m-n-1} values, and k takes on 2^n values. In this part of the code, there is really just one command involving a multiplication, namely, the one in which v is computed. This command will be encountered a number of times equal to the product $m \times 2^{m-n-1} \times 2^n = m2^{m-1}$. At an earlier point in the code, the computation of the Z-array involves $2^n - 1$ multiplications. On any binary computer, a multiplication by $1/2$ need not be counted because it is accomplished by subtracting 1 from the exponent of the floating-point number. Therefore, the total number of multiplications used in `CODE FFT.1` is

$$m2^{m-1} + 2^m - 1 \leqslant m2^m = N \log_2 N.$$

Fast inverse Fourier transform

The fast Fourier transform can also be used to evaluate the inverse transform:

$$d_k = \frac{1}{N}\sum_{j=0}^{N-1} g(x_j)e^{ikx_j}, \qquad 0 \leqslant k \leqslant N-1.$$

Let $j = N-1-m$. It is easy to verify that

$$d_k = e^{-ix_k}\frac{1}{N}\sum_{m=0}^{N-1} g(x_{N-1-m})e^{-ikx_m}, \qquad 0 \leqslant k \leqslant N-1.$$

Thus, we apply the FFT algorithm to get $e^{ix_k}d_k$. Then extra N operations give d_k. A pseudocode for computing d_k is given below.

```
CODE FFT.2
% Fast Inverse Fourier Transform
Input m
N=2^m, w=e^{-2πi/N}
for k=0 to N-1 do
    Z(k)=w^k,  C(k)=g(2π(N-1-k)/N)
end
For n=0 to m-1 do
    for k=0 to 2^{m-n-1}-1 do
        for j=0 to 2^n-1 do
            u=C(2^n k+j); v=Z(j2^{m-n-1})*C(2^n k+2^{m-1}+j)
            D(2^{n+1}k+j)=0.5*(u+v); D(2^{n+1}k+j+2^n)=0.5*(u-v)
        endfor
    endfor
    for j=0 to N-1 do
        C(j)=D(j)
    endfor
endFor
for k=0 to N-1 do
    D(k)=Z(k)*C(k)
endfor
Output D(0), D(1), ···, D(N-1).
```

Fast Cosine transform

The fast Fourier transform can also be used to evaluate the cosine transform:

$$a_k = \sum_{j=0}^{N} f(x_j)\cos\left(\pi jk/N\right), \qquad 0 \leqslant k \leqslant N,$$

where the $f(x_j)$ are *real numbers*. Let $v_j = f(x_j)$ for $0 \leqslant j \leqslant N$ and $v_j = 0$ for $N+1 \leqslant j \leqslant 2N-1$. We compute

$$A_k = \frac{1}{2N} \sum_{j=0}^{2N-1} v_j e^{-ikx_j}, \qquad x_j = \frac{\pi j}{N}, \quad 0 \leqslant k \leqslant 2N-1.$$

Since the v_j are real numbers and $v_j = 0$ for $j \geqslant N+1$, it can be shown that the real part of A_k is

$$\mathrm{Re}(A_k) = \frac{1}{2N} \sum_{j=0}^{N} f(x_j) \cos(\pi jk/N), \qquad 0 \leqslant k \leqslant 2N-1.$$

In other words, the following results hold: $a_k = 2N\mathrm{Re}(A_k)$, $0 \leqslant k \leqslant N$. By the definition of the A_k, we know that they can be computed by using the pseudocode FFT.1. When they are multiplied by $2N$, we have the values of a_k.

Numerical examples

To test the the efficiency of the FFT algorithm, we compute the coefficients in (1.5.6) using `CODE FFT.1` and the direct method. A subroutine for computing the coefficients directly from the formulas goes as follows:

```
CODE FFT.3
% Direct method for computing the coefficients
Input  m
N=2^m, w=e^{-2πi/N}
for k=0 to N-1 do
    Z(k)=w^k, D(k)=f(2πk/N)
endfor
for n=0 to N-1 do
    u=D(0)+Σ_{k=1}^{N-1} D(k)*Z(n)^k
    C(n)=u/N
endfor
Output  C(0), C(1), ···, C(N-1)
```

The computer programs based on `CODE FFT.1` and `CODE FFT.2` are written in FORTRAN with double precision. We compute the following coefficients:

$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} \cos(5x_j) e^{-ikx_j}, \qquad 0 \leqslant k \leqslant N-1,$$

where $x_j = 2\pi j/N$. The CPU time used are listed in the following table.

m	N	CPU (FFT)	CPU(direct)
9	512	0.02	0.5
10	1024	0.04	2.1
11	2048	0.12	9.0
12	4096	0.28	41.0
13	8192	0.60	180.0

The discrete Fourier transform

Again let f be a 2π-periodic function defined in $[0, 2\pi]$. The Fourier transform of $f(t)$ is defined as

$$H(s) = \mathcal{F}\{f(t)\} = \frac{1}{2\pi}\int_0^{2\pi} f(t)e^{-ist}\mathrm{d}t, \tag{1.5.13}$$

where s is a real parameter and $\mathcal{F}$ is called the Fourier transform operator. The inverse Fourier transform is denoted by $\mathcal{F}^{-1}\{H(s)\}$,

$$f(t) = \mathcal{F}^{-1}\{H(s)\} = \int_{-\infty}^{\infty} e^{ist}H(s)\mathrm{d}s,$$

where $\mathcal{F}^{-1}$ is called the inverse Fourier transform operator. The following result is important: The Fourier transform operator $\mathcal{F}$ is a linear operator satisfying

$$\mathcal{F}\{f^{(n)}(t)\} = (ik)^n\mathcal{F}\{f(t)\}, \tag{1.5.14}$$

where $f^{(n)}(t)$ denotes the n-th order derivative of $f(t)$. Similar to the continuous Fourier transform, we will define the discrete Fourier transform below. Let the solution interval be $[0, 2\pi]$. We first transform $u(x, t)$ into the discrete Fourier space:

$$\hat{u}(k, t) = \frac{1}{N}\sum_{j=0}^{N-1} u(x_j, t)e^{-ikx_j}, \qquad -\frac{N}{2} \leqslant k \leqslant \frac{N}{2} - 1, \tag{1.5.15}$$

where $x_j = 2\pi j/N$. Due to the orthogonality relation (1.5.4),

$$\frac{1}{N}\sum_{j=0}^{N-1} e^{ipx_j} = \begin{cases} 1 & \text{if } p = Nm, m = 0, \pm1, \pm2, \cdots, \\ 0 & \text{otherwise}, \end{cases}$$

we have the inversion formula

$$u(x_j,t)=\sum_{k=-N/2}^{N/2-1}\hat{u}(k,t)e^{ikx_j},\qquad 0\leqslant j\leqslant N-1. \tag{1.5.16}$$

We close this section by pointing out there are many useful developments on fast transforms by following similar spirits of the FFT methods; see e.g. [124], [126], [2], [150], [21], [65], [123], [143].

Exercise 1.5

Problem 1 Prove (1.5.4).

Problem 2 One of the most important uses of the FFT algorithm is that it allows periodic discrete convolutions of vectors of length n to be performed in $\mathcal{O}(n\log n)$ operations.

To keep the notation simple, let us consider $n=4$ (the proof below carries through in just the same way for any size). Use the fact that

$$\begin{bmatrix}1&1&1&1\\1&\omega&\omega^2&\omega^3\\1&\omega^2&\omega^4&\omega^6\\1&\omega^3&\omega^6&\omega^9\end{bmatrix}\begin{bmatrix}\hat{u}_0\\\hat{u}_1\\\hat{u}_2\\\hat{u}_3\end{bmatrix}=\begin{bmatrix}u_0\\u_1\\u_2\\u_3\end{bmatrix},$$

is quivalent to

$$\frac{1}{n}\begin{bmatrix}1&1&1&1\\1&\omega^{-1}&\omega^{-2}&\omega^{-3}\\1&\omega^{-2}&\omega^{-4}&\omega^{-6}\\1&\omega^{-3}&\omega^{-6}&\omega^{-9}\end{bmatrix}\begin{bmatrix}u_0\\u_1\\u_2\\u_3\end{bmatrix}=\begin{bmatrix}\hat{u}_0\\\hat{u}_1\\\hat{u}_2\\\hat{u}_3\end{bmatrix},$$

where $\omega=e^{2\pi i/n}$, prove that the linear system

$$\begin{bmatrix}z_0&z_3&z_2&z_1\\z_1&z_0&z_3&z_2\\z_2&z_1&z_0&z_3\\z_3&z_2&z_1&z_0\end{bmatrix}\begin{bmatrix}x_0\\x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}y_0\\y_1\\y_2\\y_3\end{bmatrix}$$

where $\{z_0,z_1,z_2,z_3\}$ is an arbitrary vector, can be transformed to a simple system of

the form

$$\begin{bmatrix} \hat{z}_0 & & & \\ & \hat{z}_1 & & \\ & & \hat{z}_2 & \\ & & & \hat{z}_3 \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \frac{1}{n} \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}.$$

1.6 Several popular time discretization methods

General Runge-Kutta methods
Stability of Runge-Kutta methods
Multistep methods
Backward difference methods (BDF)
Operator splitting methods

We present in this section several popular time discretization methods, which will be repeatedly used in this book, for a system of ordinary differential equations

$$\frac{dU}{dt} = F(U,t), \tag{1.6.1}$$

where $U \in \mathbb{R}^d, F \in \mathbb{R}^d$. An initial condition is also given to the above problem:

$$U(t_0) = U_0. \tag{1.6.2}$$

The simplest method is to approximate dU/dt by the finite difference quotient $U'(t)$ $\approx [U(t+\Delta t) - U(t)]/\Delta t$. Since the starting data is known from the initial condition $U^0 = U_0$, we can obtain an approximation to the solution at $t_1 = t_0 + \Delta t$: $U^1 = U^0 + \Delta t\, F(U^0, t_0)$. The process can be continued. Let $t_k = t_0 + k\Delta t$, $k \geqslant 1$. Then the approximation U_{k+1} to the solution $U(t_{k+1})$ is given by

$$U^{n+1} = U^n + \Delta t\, F(U^n, t_n), \tag{1.6.3}$$

where $U^n \approx U(\cdot,\ t_n)$. The above algorithm is called the *Euler method*. It is known that if the function F has a bounded partial derivative with respect to its second variable and if the solution U has a bounded second derivative, then the Euler method converges to the exact solution with first order of convergence, namely,

$$\max_{1 \leqslant n \leqslant N} |U^n - U(t_n)| \leqslant C\Delta t,$$

where C is independent of N and Δt.

The conceptually simplest approach to higher-order methods is to use more terms in the Taylor expansion. Compared with the Euler method, one more term is taken so that

$$U(t_{n+1}) \approx U(t_n) + \Delta t\, U'(t_n) + \frac{\Delta t^2}{2} U''(t_n), \tag{1.6.4}$$

where the remainder of $\mathcal{O}(\Delta t^3)$ has been dropped. It follows from (1.6.1) that $U'(t_n)$ can be replaced by $F(U^n, t_n)$. Moreover,

$$U''(t) = \frac{d}{dt} F(U(t), t) = F_U(U, t) U'(t) + F_t(U, t)\,,$$

which yields

$$U''(t_n) \approx F_U(U^n, t_n) F(U^n, t_n) + F_t(U^n, t_n).$$

Using this to replace $U''(t_n)$ in (1.6.4) leads to the method

$$U^{n+1} = U^n + \Delta t F(U^n, t_n) + \frac{\Delta t^2}{2} [F_t(U^n, t_n) + F_U(U^n, t_n) F(U^n, t_n)]. \tag{1.6.5}$$

It can be shown the above scheme has second-order order accuracy provided that F and the underlying solution U are smooth.

General Runge-Kutta methods

Instead of computing the partial derivatives of F, we could also obtain higher-order methods by making more evaluations of the function values of F at each step. A class of such schemes is known as Runge-Kutta methods. The second-order Runge-Kutta method is of the form:

$$\begin{cases} U = U^n, & G = F(U, t_n), \\ U = U + \alpha \Delta t G, & G = (-1 + 2\alpha - 2\alpha^2) G + F(U, t_n + \alpha \Delta t), \\ U^{n+1} = U + \dfrac{\Delta t}{2\alpha} G. \end{cases} \tag{1.6.6}$$

Only two levels of storage (U and G) are required for the above algorithm. The choice $\alpha = 1/2$ produces the modified Euler method, and $\alpha = 1$ corresponds to the Heun method.

The third-order Runge-Kutta method is given by:

$$\begin{cases} U = U^n, & G = F(U, t_n), \\ U = U + \dfrac{1}{3}\Delta t G, & G = -\dfrac{5}{9}G + F\left(U, t_n + \dfrac{1}{3}\Delta t\right), \\ U = U + \dfrac{15}{16}\Delta t G, & G = -\dfrac{153}{128}G + F\left(U, t_n + \dfrac{3}{4}\Delta t\right), \\ U^{n+1} = U + \dfrac{8}{15}G. \end{cases} \tag{1.6.7}$$

Only two levels of storage (U and G) are required for the above algorithm.

The classical fourth-order Runge-Kutta (RK4) method is

$$\begin{cases} K_1 = F(U^n, t_n), & K_2 = F\left(U^n + \dfrac{\Delta t}{2}K_1, t_n + \dfrac{1}{2}\Delta t\right), \\ K_3 = F\left(U^n + \dfrac{\Delta t}{2}K_2, t_n + \dfrac{1}{2}\Delta t\right), & K_4 = F(U^n + \Delta t K_3, t_{n+1}), \\ U^{n+1} = U^n + \dfrac{\Delta t}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right). \end{cases} \tag{1.6.8}$$

The above formula requires four levels of storage, i.e. K_1, K_2, K_3 and K_4. An equivalent formulation is

$$\begin{cases} U = U^n, & G = U, \quad P = F(U, t_n), \\ U = U + \dfrac{1}{2}\Delta t P, & G = P, \quad P = F\left(U, t_n + \dfrac{1}{2}\Delta t\right), \\ U = U + \dfrac{1}{2}\Delta t(P - G), & G = \dfrac{1}{6}G, \quad P = F\left(U, t_n + \Delta t/2\right) - P/2, \\ U = U + \Delta t P, & G = G - P, \quad P = F(U, t_{n+1}) + 2P, \\ U^{n+1} = U + \Delta t\left(G + P/6\right). \end{cases} \tag{1.6.9}$$

This version of the RK4 method requires only three levels (U, G and P) of storage.

As we saw in the derivation of the Runge-Kutta method of order 2, a number of parameters must be selected. A similar process occurs in establishing higher-order Runge-Kutta methods. Consequently, there is not just one Runge-Kutta method for each order, but a family of methods. As shown in the following table, the number of required *function evaluations* increases more rapidly than the order of the Runge-Kutta methods:

Number of function evaluations	1	2	3	4	5	6	7	8
Maximum order of RK method	1	2	3	4	4	5	6	6

Unfortunately, this makes the higher-order Runge-Kutta methods less attractive than the classical fourth-order method, since they are more expensive to use.

The Runge-Kutta procedure for systems of first-order equations is most easily written down in the case when the system is *autonomous*; that is, it has the form

$$\frac{dU}{dt} = F(U). \tag{1.6.10}$$

The classical RK4 formulas, in vector form, are

$$U^{n+1} = U^n + \frac{\Delta t}{6}\Big(K_1 + 2K_2 + 2K_3 + K_4\Big), \tag{1.6.11}$$

where

$$\begin{cases} K_1 = F(U^n), \quad K_2 = F\left(U^n + \dfrac{\Delta t}{2}K_1\right), \\ K_3 = F\left(U^n + \dfrac{\Delta t}{2}K_2\right), \quad K_4 = F\left(U^n + \Delta t K_3\right). \end{cases}$$

For problems without source terms such as Examples 5.3.1 and 5.3.2, we will end up with an autonomous system. The above RK4 method, or its equivalent form similar to (1.6.9), can be used.

Stability of Runge-Kutta methods

The general s-stage explicit Runge-Kutta method of maximum order s has stability function

$$r(z) = 1 + z + \frac{z^2}{2} + \cdots + \frac{z^s}{s!}, \quad s = 1, 2, 3, 4. \tag{1.6.12}$$

There are a few stability concepts for the Runge-Kutta methods:

a. The region of absolute stability $\mathcal{R}$ of an s-order Runge-Kutta method is the set of points $z = \lambda\Delta t \in \mathbb{C}$ such that if $z \in \mathcal{R}$, $(\mathcal{R}e(\lambda) < 0)$. Then the numerical method applied to

$$\frac{du}{dt} = \lambda u \tag{1.6.13}$$

gives $u^n \to 0$ as $n \to \infty$. It can be shown that the region of absolute stability of a

Runge-Kutta method is given by

$$\mathcal{R} = \{z \in \mathbb{C} \mid |r(z)| < 1\}. \tag{1.6.14}$$

b. A Runge-Kutta method is said to be A-stable if its stability region contains the left-half of the complex plane, i.e. the non-positive half-plane, $\mathbb{C}^-$.

c. A Runge-Kutta method is said to be L-stable if it is A-stable, and if its stability function $r(z)$ satisfies

$$\lim_{|z|\to\infty} |r(z)| = 0. \tag{1.6.15}$$

In Figure 1.3, we can see that the stability domains for these explicit Runge-Kutta methods consist of the interior of closed regions in the left-half of the complex plane. The algorithm for plotting the absolute stability regions above can be found in the book by Butcher [27]. Notice that all Runge-Kutta methods of a given order have the same stability properties. The stability regions expand as the order increases.

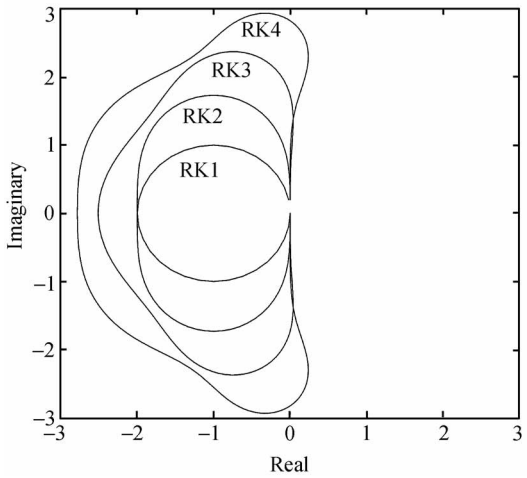

Figure 1.3 Absolute stability regions of Runge-Kutta methods

Multistep methods

Another approach to higher-order methods utilizes information already computed and does not require additional evaluations of $F(U,t)$. One of the simplest such methods is

$$U^{n+1} = U_n + \frac{\Delta t}{2}[3F(U^n, t_n) - F(U^{n-1}, t_{n-1})], \tag{1.6.16}$$

for which the maximum pointwise error is $\mathcal{O}(\Delta t^2)$, and is known as the second-order

Adams-Bashforth method, or AB2 for short. Note that the method requires only the evaluation of $F(U^n, t_n)$ at each step, the value $F(U^{n-1}, t_{n-1})$ being known from the previous step.

We now consider the general construction of Adams-Bashforth methods. Let $U^n, U^{n-1}, \cdots, U^{n-s}$ be the computed approximations to the solution at t_n, t_{n-1}, $\cdots, t_{n-s}$. Let $F^i = F(U^i, t_i)$ and let $p(t)$ be the interpolating polynomial of degree s that satisfies

$$p(t_i) = F^i, \qquad i = n, n-1, \cdots, n-s.$$

We may then consider $p(t)$ to be an approximation to $F(U(t), t)$. Since the solution $U(t)$ satisfies

$$U(t_{n+1}) - U(t_n) = \int_{t_n}^{t_{n+1}} U'(t)\mathrm{d}t = \int_{t_n}^{t_{n+1}} F(U(t), t)\mathrm{d}t \approx \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t,$$

we obtain the so-called Adams-Bashforth (AB) methods as follows:

$$U^{n+1} = U^n + \int_{t_n}^{t_{n+1}} p(t)\mathrm{d}t. \tag{1.6.17}$$

Below we provide a few special cases of the Adams-Bashforth methods:

- $s = 0$: $p(t) = F_n$ for $t \in [t_n, t_{n+1})$, gives Euler method.
- $s = 1$:

$$p(t) = p_1(t) = U^n + \frac{t - t_n}{\Delta t}(F^n - F^{n-1}),$$

which leads to the second-order Adams-Bashforth method (1.6.16).

- $s = 2$:

$$p_2(t) = p_1(t) + \frac{(t - t_n)(t - t_{n-1})}{2\Delta t^2}(F^n - 2F^{n-1} + F^{n-2}),$$

which leads to the third-order Adams-Bashforth method

$$U^{n+1} = U^n + \frac{\Delta t}{12}(23F^n - 16F^{n-1} + 5F^{n-2}). \tag{1.6.18}$$

- $s = 3$:

$$p_3(t) = p_2(t) - \frac{(t - t_n)(t - t_{n-1})(t - t_{n-2})}{3!\Delta t^3}(F^n - 3F^{n-1} + 3F^{n-2} - F^{n-3}),$$

which leads to the fourth-order Adams-Bashforth method

$$U^{n+1} = U^n + \frac{\Delta t}{24}(55F^n - 59F^{n-1} + 37F^{n-2} - 9F^{n-3}). \tag{1.6.19}$$

In principle, we can continue the preceding process to obtain Adams-Bashforth methods of arbitrarily high-order, but the formulas become increasingly complex as d increases. The Adams-Bashforth methods are multistep methods since two or more levels of prior data are used. This is in contrast to the Runge-Kutta methods which use no prior data and are called one-step methods. We will compute the numerical solutions of the KdV equation using a multistep method (see Sect. 5.4).

Multistep methods cannot start by themselves. For example, consider the fourth-order Adams-Bashforth method. The initial value U^0 is given, but for $k = 0$, the information is needed at t_{-1}, t_{-2}, t_{-3}, which is not available. The method needs "help" getting started. We cannot use the fourth-order multistep method until $k \geqslant 3$. A common policy is to use a one-step method, such as a Runge-Kutta method of the same order of accuracy at some starting steps.

Since the Adams-Bashforth methods of arbitrary order require only one evaluation of $F(U, t)$ at each step, the "cost" is lower than that of Runge-Kutta methods. On the other hand, in Runge-Kutta methods it is much easier to change step-size; hence they are more suitable for use in an adaptive algorithm.

Backward difference methods (BDF)

The Adams-Bashforth methods can be unstable due to the fact they are obtained by integrating the interpolating polynomial outside the interval of the data that defines the polynomial. This can be remedied by using multilevel implicit methods:

- Second-order backward difference method (BD2):

$$\frac{1}{2\Delta t}(3U^{n+1} - 4U^n + U^{n-1}) = F(U^{n+1}, t_{n+1}). \tag{1.6.20}$$

- Third-order backward difference method (BD3):

$$\frac{1}{6\Delta t}(11U^{n+1} - 18U^n + 9U^{n-1} - 2U^{n-2}) = F(U^{n+1}, t_{n+1}). \tag{1.6.21}$$

In some practical applications, $F(u, t)$ is often the sum of linear and nonlinear terms. In this case, some combination of the backward difference method and extrapolation method can be used. To fix the idea, let us consider

$$u_t = \mathcal{L}(u) + \mathcal{N}(u), \tag{1.6.22}$$

where $\mathcal{L}$ is a linear operator and $\mathcal{N}$ is a nonlinear operator. By combining a second-order backward differentiation (BD2) for the time derivative term and a second-order extrapolation (EP2) for the explicit treatment of the nonlinear term, we arrive at a second-order scheme (BD2/EP2) for (1.6.22):

$$\frac{1}{2\Delta t}(3U^{n+1} - 4U^n + U^{n-1}) = \mathcal{L}(U^{n+1}) + \mathcal{N}(2U^n - U^{n-1}). \tag{1.6.23}$$

A third-order scheme for solving (1.6.22) can be constructed in a similar manner, which leads to the so-called BD3/EP3 scheme:

$$\frac{1}{6\Delta t}(11U^{n+1} - 18U^n + 9U^{n-1} - 2U^{n-2}) = \mathcal{L}(U^{n+1}) + \mathcal{N}(3U^n - 3U^{n-1} + U^{n-2}). \tag{1.6.24}$$

Operator splitting methods

In many practical situations, $F(u,t)$ is often the sum of several terms with different properties. Then it is often advisable to use an operator splitting method (also called fractional step method)[171, 119, 57, 154]. To fix the idea, let us consider

$$u_t = f(u) = Au + Bu, \quad u(t_0) = u_0, \tag{1.6.25}$$

where $f(u)$ is a nonlinear operator and the splitting $f(u) = Au + Bu$ can be quite arbitrary; in particular, A and B do not need to commute.

`Strang's operator splitting method` For a given time step $\Delta t > 0$, let $t_n = n\ \Delta t$, $n = 0, 1, 2, \cdots$ and u^n be the approximation of $u(t_n)$. Let us formally write the solution $u(x,t)$ of (1.6.25) as

$$u(t) = e^{t(A+B)}u_0 =: S(t)u_0. \tag{1.6.26}$$

Similarly, denote by $S_1(t) := e^{tA}$ the solution operator for $u_t = Au$, and by $S_2(t) := e^{tB}$ the solution operator for $u_t = Bu$. Then the first-order operator splitting is based on the approximation

$$u^{n+1} \approx S_2(\Delta t)S_1(\Delta t)u^n, \tag{1.6.27}$$

or on the one with the roles of S_2 and S_1 reversed. To maintain second-order accuracy, the Strang splitting[154] can be used, in which the solution $S(t_n)u_0$ is approximated by

$$u^{n+1} \approx S_2(\Delta t/2)S_1(\Delta t)S_2(\Delta t/2)u^n, \tag{1.6.28}$$

or by the one with the roles of S_2 and S_1 reversed. It should be pointed out that

first-order accuracy and second-order accuracy are based on the truncation errors for smooth solutions. For discontinuous solutions, it is not difficult to show that both approximations (1.6.27) and (1.6.28) are at most first-order accurate, see e.g. [35], [159].

`Fourth-order time-splitting method` A fourth-order symplectic time integrator (cf. [172], [99]) for (1.6.25) is as follows:

$$\begin{aligned} &u^{(1)} = e^{2w_1 A\Delta t}\, u^n, \quad u^{(2)} = e^{2w_2 B\Delta t}\, u^{(1)}, \quad u^{(3)} = e^{2w_3 A\Delta t}\, u^{(2)}, \\ &u^{(4)} = e^{2w_4 B\Delta t}\, u^{(3)}, \quad u^{(5)} = e^{2w_3 A\Delta t}\, u^{(4)}, \quad u^{(6)} = e^{2w_2 B\Delta t}\, u^{(5)}, \\ &u^{n+1} = e^{2w_1 A\Delta t}\, u^{(6)}; \end{aligned} \tag{1.6.29}$$

or, equivalently,

$$\begin{aligned} u^{n+1} \approx & S_1(2w_1\Delta t)S_2(2w_2\Delta t)S_1(2w_3\Delta t)S_2(2w_4\Delta t) \\ & S_1(2w_3\Delta t)S_2(2w_2\Delta t)S_1(2w_1\Delta t)u^n, \end{aligned}$$

where

$$\begin{aligned} &w_1 = 0.33780\ 17979\ 89914\ 40851,\ w_2 = 0.67560\ 35959\ 79828\ 81702, \\ &w_3 = -0.08780\ 17979\ 89914\ 40851,\ w_4 = -0.85120\ 71979\ 59657\ 63405. \end{aligned} \tag{1.6.30}$$

Numerical tests

To test the Runge-Kutta algorithms discussed above, we consider Example 5.3.1 in Section 5.3. Let $U = (U_1, \cdots, U_{N-1})^{\mathrm{T}}$, namely the vector of approximation values at the interior Chebyshev points. Using the definition of the differentiation matrix to be provided in the next chapter, the Chebyshev pesudospectral method for the heat equation (1.1.1) with homogeneous boundary condition leads to the system

$$\frac{dU}{dt} = AU,$$

where A is a constant matrix with $(A)_{ij} = (D^2)_{ij}$. The matrix $D^2 = D^1 * D^1$, where D^1 is given by `CODE DM.3` in Sect 2.1. The following pseudo-code implements the RK2 (1.6.6).

```
CODE RK.1
Input N, u_0(x), Δt, Tmax, α
%Form the matrix A
```

```
call CODE DM.3 in Sect 2.1 to get D1(i,j), 0⩽i,j⩽N
D2=D1*D1;
A(i,j)=D2(i,j), 1⩽i,j⩽N-1
Set starting time:  time=0
Set the initial data:  U0=u_0(x)
While time⩽Tmax do
   %Using RK2 (1.6.6)
   U=U0; G=A*U
   U=U+α*Δt*G; G=(-1+2α-2α^2)G+A*U
   U0=U+Δt*G/(2*α)
   Set new time level:  time=time+Δt
endWhile
Output U0(1),U(2), ···, U(N-1)
```

Codes using (1.6.11), i.e., RK4 for autonomous system, can be written in a similar way. Numerical results for Example 5.3.1 using RK2 with $\alpha = 1$ (i.e., the Heun method) and RK4 are given in the following table. `Tmax` in the above code is set to be 0.5. It is seen that these results are more accurate than the forward Euler solutions obtained in Section 5.3.

N	Heun method ($\Delta t=10^{-3}$)	RK4 ($\Delta t=10^{-3}$)
3	1.11e-02	1.11e-02
4	3.75e-03	3.75e-03
6	3.99e-05	4.05e-05
8	1.23e-06	1.77e-06
10	5.92e-07	3.37e-08
11	5.59e-07	1.43e-09
12	5.80e-07	4.32e-10

The numerical errors for $\Delta t = 10^{-3}$, Tmax=0.5 and different values of s (the order of accuracy) can be seen from the following table:

N	s=2	s=3	s=4
3	1.11e-02	1.11e-02	1.11e-02
4	3.75e-03	3.75e-03	3.75e-03
6	3.99e-05	4.05e-05	4.05e-05
8	1.23e-06	1.77e-06	1.77e-06
10	5.92e-07	3.23e-08	3.37e-08
11	5.59e-07	2.82e-09	1.43e-09
12	5.80e-07	1.70e-09	4.32e-10

Exercise 1.6

Problem 1 Solve the problem in Example 5.3.1 by using a pseudo-spectral ap-

proach (i.e. using the differential matrix to solve the problem in the physical space). Take $3 \leqslant N \leqslant 20$, and use RK4.

1.7 Iterative methods and preconditioning

BiCG algorithm
CGS algorithm
BiCGSTAB algorithm
GMRES method
Preconditioning techniques
Preconditioned GMRES

Among the iterative methods developed for solving large sparse problems, we will mainly discuss two methods: the conjugate gradient (CG) method and the generalized minimal residual (GMRES) method. The CG method proposed by Hestenes and Stiefel in 1952 [82] is the *method of choice* for solving large *symmetric positive definite* linear systems, while the GMRES method was proposed by Saad and Schultz in 1986 for solving non-symmetric linear systems [135].

Let the matrix $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and $b \in \mathbb{R}^n$ a given vector. It can be verified that $\hat{x}$ is the solution of $Ax = b$ if and only if $\hat{x}$ minimizes the quadratic functional

$$J(x) = \frac{1}{2}x^{\mathrm{T}}Ax - x^{\mathrm{T}}b. \tag{1.7.1}$$

Let us consider the minimization procedure. Suppose x_k has been obtained. Then x_{k+1} can be found by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{1.7.2}$$

where the scalar α_k is called the step size factor and the vector p_k is called the search direction. The coefficient α_k in (1.7.2) is selected such that $J(x_k + \alpha_k p_k) = \min_\alpha J(x_k + \alpha p_k)$. A simple calculation shows that

$$\alpha_k = (r_k, p_k)/(Ap_k, p_k) = p_k^{\mathrm{T}} r_k / p_k^{\mathrm{T}} A p_k.$$

The residual at this step is given by

$$\begin{aligned} r_{k+1} &= b - Ax_{k+1} = b - A(x_k + \alpha_k p_k) \\ &= b - Ax_k - \alpha_k Ap_k = r_k - \alpha_k Ap_k. \end{aligned}$$

Select the *next search direction* p_{k+1} such that $(p_{k+1}, Ap_k) = 0$, i.e,

$$p_{k+1} = r_{k+1} + \beta_k p_k, \tag{1.7.3}$$

where

$$\beta_k = -\frac{(Ap_k, r_{k+1})}{(Ap_k, p_k)} = -\frac{r_{k+1}^{\mathrm{T}} A p_k}{p_k^{\mathrm{T}} A p_k}.$$

It can be verified that

$$r_i^{\mathrm{T}} r_j = 0, \quad p_i^{\mathrm{T}} A p_j = 0, \quad i \neq j. \tag{1.7.4}$$

Consequently, it can be shown that if A is a real $n \times n$ symmetric positive definite matrix, then the iteration converges in at most n steps, i.e. $x_m = \hat{x}$ for some $m \leqslant n$.

The above derivations lead to the following conjugate gradient (CG) algorithm:

Choose x_0, compute $r_0 = b - Ax_0$ and set $p_0 = r_0$.
For $k = 0, 1, \cdots$ *do*
Compute $\alpha_k = (r_k, r_k)/(Ap_k, p_k)$
  Set $x_{k+1} = x_k + \alpha_k p_k$
  Compute $r_{k+1} = r_k - \alpha_k A p_k$
  If $\|r_{k+1}\|_2 \geqslant \epsilon$, continue,
  Compute $\beta_k = (r_{k+1}, r_{k+1})/(r_k, r_k)$
  Set $p_{k+1} = r_{k+1} + \beta_k p_k$
endFor

It is left as an exercise for the reader to prove that these coefficient formulas in the CG algorithm are equivalent to the obvious expressions in the above derivations.

The rate of convergence of the conjugate gradient method is given by the following theorem:

Theorem 1.7.1 *If A is a symmetric positive definite matrix, then the error of the conjugate gradient method satisfies*

$$\|\hat{x} - x_k\|_A \leqslant 2\gamma^k \|\hat{x} - x_0\|_A, \tag{1.7.5}$$

where

$$\|x\|_A = (Ax, x) = x^{\mathrm{T}} A x, \quad \gamma = (\sqrt{\kappa} - 1)/(\sqrt{\kappa} + 1), \tag{1.7.6}$$

and $\kappa = \|A\|_2 \|A^{-1}\|_2$ is the condition number of A. □

For a symmetric positive definite matrix, $\|A\|_2 = \lambda_n$, $\|A^{-1}\|_2 = \lambda_1^{-1}$, where λ_n and λ_1 are the largest and smallest eigenvalues of A. It follows from Theorem 1.7.1

that a 2-norm error bound can be obtained:

$$\|\hat{x} - x_k\|_2 \leqslant 2\sqrt{\kappa}\gamma^k \|x - x_0\|_2. \tag{1.7.7}$$

We remark that

• we only have matrix-vector multiplications in the CG algorithm. In case that the matrix is sparse or has a special structure, these multiplications can be done efficiently.

• unlike the traditional successive over-relaxation (SOR) type method, there is no free parameter to choose in the CG algorithm.

BiCG algorithms

When the matrix A is non-symmetric, an direct extension of the CG algorithm is the so called biconjugate gradient (BiCG) method.

The BiCG method aims to solve $Ax = b$ and $A^{\mathrm{T}}x^* = b^*$ simultaneously. The iterative solutions are updated by

$$x_{j+1} = x_j + \alpha_j p_j, \qquad x^*_{j+1} = x^*_j + \alpha_j p^*_j \tag{1.7.8}$$

and so

$$r_{j+1} = r_j - \alpha_j A p_j, \qquad r^*_{j+1} = r^*_j - \alpha_j A^{\mathrm{T}} p^*_j. \tag{1.7.9}$$

We require that $(r_{j+1}, r^*_j) = 0$ and $(r_j, r^*_{j+1}) = 0$ for all j. This leads to

$$\alpha_j = (r_j, r^*_j)/(Ap_j, p^*_j). \tag{1.7.10}$$

The search directions are updated by

$$p_{j+1} = r_{j+1} + \beta_j p_j, \qquad p^*_{j+1} = r^*_{j+1} + \beta_j p^*_j. \tag{1.7.11}$$

By requiring that $(Ap_{j+1}, p^*_j) = 0$ and $(Ap_j, p^*_{j+1}) = 0$, we obtain

$$\beta_j = (r_{j+1}, r^*_{j+1})/(r_j, r^*_j). \tag{1.7.12}$$

The above derivations lead to the following BiCG algorithm:

Choose x_0, compute $r_0 = b - Ax_0$ and set $p_0 = r_0$.
Choose r^*_0 such that $(r_0, r^*_0) \neq 0$.
For $j = 0, 1, \cdots$ do

Compute $\alpha_j = \frac{(r_j, r_j^*)}{(Ap_j, p_j^*)}$.
Set $x_{j+1} = x_j + \alpha_j p_j$.
Compute $r_{j+1} = r_j - \alpha_j A p_j$ and $r_{j+1}^* = r_j^* - \alpha_j A^{\mathrm{T}} p_j^*$.
If $\|r_{k+1}\|_2 \geqslant \epsilon$, continue,
Compute $\beta_j = \frac{(r_{j+1}, r_{j+1}^*)}{(r_j, r_j^*)}$.
Set $p_{j+1} = r_{j+1} + \beta_j p_j$ and $p_{j+1}^* = r_{j+1}^* + \beta_j p_j^*$
endFor

We remark that

• The BiCG algorithm is particularly suitable for matrices which are positive definite, i.e., $(Ax, x) > 0$ for all $x \neq 0$, but not symmetric.

• the algorithm breaks down if $(Ap_j, p_j^*) = 0$. Otherwise, the amount of work and storage is of the same order as n the CG algorithm.

• if A is symmetric and $r_0^* = r_0$, then the BiCG algorithm reduces to the CG algorithm.

CGS algorithm

The BiCG algorithm requires multiplication by both A and A^{T} at each step. Obviously, this means extra work, and, additionally, it is sometimes cumbersome to multiply by A^{T} than it is to multiply by A. For example, there may be a special formula for the product of A with a given vector when A represents, say, a Jacobian, but a corresponding formula for the product of A^{T} with a given vector may not be available. In other cases, data may be stored on a parallel machine in such a way that multiplication by A is efficient but multiplication by A^{T} involves extra communication between processors. For these reasons it is desirable to have an iterative method that requires multiplication only by A and that generates good approximate solutions. A method that attempts to do this is the conjugate gradient squared (CGS) method.

For the recurrence relations of BiCG algorithms, we see that

$$r_j = \Phi_j^1(A) r_0 + \Phi_j^2(A) p_0,$$

where $\Phi_j^1(A)$ and $\Phi_j^2(A)$ are j-th order polynomials of the matrix A. Choosing $p_0 = r_0$ gives

$$r_j = \Phi_j(A) r_0 \qquad (\Phi_j = \Phi_j^1 + \Phi_j^2),$$

with $\Phi_0 \equiv 1$. Similarly,

$$p_j = \pi_j(A)r_0,$$

where π_j is a polynomial of degree j. As r_j^* and p_j^* are updated, using the same recurrence relation as for r_j and p_j, we have

$$r_j^* = \Phi_j(A^{\mathrm{T}})r_0^*, \qquad p_j^* = \pi_j(A^{\mathrm{T}})r_0^*. \tag{1.7.13}$$

Hence,

$$\alpha_j = \frac{(\Phi_j(A)r_0, \Phi_j(A^{\mathrm{T}})r_0^*)}{(A\pi_j(A)r_0, \pi_j(A^{\mathrm{T}})r_0^*)} = \frac{(\Phi_j^2(A)r_0, r_0^*)}{(A\pi_j^2(A)r_0, r_0^*)}\,. \tag{1.7.14}$$

From the BiCG algorithm:

$$\Phi_{j+1}(t) = \Phi_j(t) - \alpha_j t\pi_j(t), \quad \pi_{j+1}(t) = \Phi_{j+1}(t) + \beta_j\pi_j(t). \tag{1.7.15}$$

Observe that

$$\Phi_j\pi_j = \Phi_j(\Phi_j + \beta_{j-1}\pi_{j-1}) = \Phi_j^2 + \beta_{j-1}\Phi_j\pi_{j-1}. \tag{1.7.16}$$

It follows from the above results that

$$\Phi_{j+1}^2 = \Phi_j^2 - 2\alpha_j t(\Phi_j^2 + \beta_{j-1}\Phi_j\pi_{j-1}) + \alpha_j^2 t^2\pi_j^2,$$

$$\Phi_{j+1}\pi_j = \Phi_j\pi_j - \alpha_j t\pi_j^2 = \Phi_j^2 + \beta_{j-1}\Phi_j\pi_{j-1} - \alpha_j t\pi_j^2,$$

$$\pi_{j+1}^2 = \Phi_{j+1}^2 + 2\beta_j\Phi_{j+1}\pi_j + \beta_j^2\pi_j^2.$$

Define

$$\begin{aligned} r_j &= \Phi_j^2(A)r_0, \quad p_j = \pi_j^2(A)r_0,\\ q_j &= \Phi_{j+1}(A)\pi_j(A)r_0,\\ d_j &= 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j Ap_j. \end{aligned}$$

It can be verified that

$$\begin{aligned} r_j &= r_{j-1} - \alpha_j Ad_j,\\ q_j &= r_j + \beta_{j-1}q_{j-1} - \alpha_j Ap_j\\ p_{j+1} &= r_{j+1} + 2\beta_j q_j + \beta_j^2 p_j,\\ d_j &= 2r_j + 2\beta_{j-1}q_{j-1} - \alpha_j Ap_j. \end{aligned}$$

Correspondingly,

$$x_{j+1} = x_j + \alpha_j d_j. \tag{1.7.17}$$

This gives the CGS algorithm. It is true that x_j may not be the same as that produced by the BiCG.

The above derivations lead to the following the CGS algorithm:

Choose x_0, compute $r_0 = b - Ax_0$ and set $p_0 = r_0, u_0 = r_0,\ q_0 = 0$.
Choose r_0^* such that $(r_0, r_0^*) \neq 0$.
For $j = 0, 1, \cdots$ do
  Compute $\alpha_j = \frac{(r_j, r_0^*)}{(Ap_j, r_0^*)}$; Compute $q_{j+1} = u_j - \alpha_j A p_j$
  Set $x_{j+1} = x_j + \alpha_j(u_j + q_{j+1})$
  Compute $r_{j+1} = r_j - \alpha_j A(u_j + q_{j+1})$
  If $\|r_{k+1}\|_2 \geqslant \epsilon$, continue,
  Compute $\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)}$; Compute $u_{j+1} = r_{j+1} + \beta_j q_{j+1}$
  Set $p_{j+1} = u_{j+1} + \beta_j(q_{j+1} + \beta_j p_j)$
endFor

The CGS method requires two matrix-vector multiplications at each step but no multiplications by the transpose. For problems where the BiCG method converges well, the CGS typically requires only about half as many steps and, therefore, half the work of BiCG (assuming that multiplication by A or A^{T} requires the same amount of work). When the norm of the BiCG residual increases at a step, however, that of the CGS residual usually increases by approximately the square of the increase of the BiCG residual norm. The CGS convergence curve may therefore show wild oscillations that can sometimes lead to numerical instabilities.

BiCGSTAB algorithm

To avoid the large oscillations in the CGS convergence curve, one might try to produce a residual of the form

$$r_j = \Psi_j(A)\Phi_j(A)r_0, \tag{1.7.18}$$

where Φ_j is again the BiCG polynomial but Ψ_j is chosen to keep the residual norm small at each step while retaining the rapid overall convergence of the CGS method.

For example, $\Psi_j(t)$ is of the form

$$\Psi_{j+1}(t) = (1 - w_j t)\Psi_j(t). \tag{1.7.19}$$

In the BiCGSTAB algorithm, the solution is updated in such a way that r_j is of the form (1.7.18), where $\Psi_j(A)$ is a polynomial of degree j which satisfies (1.7.19). It can be shown that

$$\begin{aligned} \Psi_{j+1}\Phi_{j+1} &= (1 - w_j t)\Psi_j(\Phi_j - \alpha_j t\pi_j) \\ &= (1 - w_j t)(\Psi_j\Phi_j - \alpha_j t\Psi_j\pi_j), \end{aligned} \tag{1.7.20}$$

$$\begin{aligned} \Psi_j\pi_j &= \Psi_j(\Phi_j + \beta_{j-1}\pi_{j-1}) \\ &= \Psi_j\Phi_j + \beta_{j-1}(1 - w_{j-1}t)\Psi_{j-1}\pi_{j-1}. \end{aligned} \tag{1.7.21}$$

Let $r_j = \Phi_j(A)\Psi_j(A)r_0$ and $p_j = \Psi_j(A)\pi_j(A)r_0$. It can be verified that

$$\begin{aligned} r_{j+1} &= (I - w_j A)(r_j - \alpha_j A p_j), \\ p_{j+1} &= r_{j+1} + \beta_j(I - w_j A)p_j. \end{aligned} \tag{1.7.22}$$

By letting $s_j = r_j - \alpha_j A p_j$, we obtain

$$r_{j+1} = (I - w_j A)s_j. \tag{1.7.23}$$

The parameter w_j is chosen to minimize the 2-norm of r_{j+1}, i.e.,

$$w_j = \frac{(As_j, s_j)}{(As_j, As_j)}. \tag{1.7.24}$$

We also need to find an updating formula for α_j and β_j, only using r_k, p_k and s_k; this is rather complicated and the calculations for deriving them are omitted here.

The BiCGSTAB algorithm is given by

Choose x_0, compute $r_0 = b - Ax_0$ and set $p_0 = r_0$.
Choose r_0^* such that $(r_0, r_0^*) \neq 0$.
For $j = 0, 1, \cdots$ do
Compute $\alpha_j = \frac{(r_j, r_0^*)}{(Ap_j, r_0^*)}$
Set $s_j = r_j - \alpha_j A p_j$; Compute $w_j = \frac{(As_j, s_j)}{(As_j, As_j)}$

Set $x_{j+1} = x_j + \alpha_j p_j + w_j s_j$; $r_{j+1} = s_j - w_j A s_j$
If $\|r_{k+1}\|_2 \geqslant \epsilon$, continue,
Compute $\beta_j = \frac{(r_{j+1}, r_0^*)}{(r_j, r_0^*)} \cdot \frac{\alpha_j}{w_j}$
Set $p_{j+1} = r_{j+1} + \beta_j (p_j - w_j A p_j)$
endFor

GMRES method

The GMRES method proposed by Saad and Schultz in 1986 is one of the most important tools for a general *non-symmetric* system

$$Ax = b, \quad \text{with } A \text{ non-symmetric}. \tag{1.7.25}$$

In the k-th iteration of the GMRES method, we need to find a solution of the least-squares problem

$$\min_{x \in x_0 + \|(A, r_0, k)} \|b - Ax\|_2 , \tag{1.7.26}$$

where $r_0 = b - Ax_0$ and $\|(A, r_0, k) := \{r_0, Ar_0, \cdots, A^{k-1} r_0\}$. Let $x \in x_0 + \|(A, r_0, k)$. We have

$$x = x_0 + \sum_{j=0}^{k-1} \gamma_j A^j r_0. \tag{1.7.27}$$

Moreover, it can be shown that

$$r = b - Ax = r_0 - \sum_{j=1}^{k} \gamma_{j-1} A^j r_0. \tag{1.7.28}$$

Like the CG method, the GMRES method will obtain the *exact* solution of $Ax = b$ within n iterations. Moreover, if b is a linear combination of k eigenvectors of A, say $b = \sum_{p=1}^{k} \gamma_p u_{i_p}$, then the GMRES method will terminate in at most k iterations.

Suppose that we have a matrix $V_k = [v_1^k, v_2^k, \cdots, v_k^k]$ whose columns form an orthogonal basis of $\|(A, r_0, k)$. Then any $z \in \|(A, r_0, k)$ can be expressed as

$$z = \sum_{p=1}^{k} u_p v_p^k = V_k u, \tag{1.7.29}$$

where $u \in \mathbb{R}^k$. Thus, once we have found V_k, we can convert the original least-squares problem (1.7.26) into a least-squares problem in $\mathbb{R}^k$, as to be described below. Let x_k be the solution after the k-th iteration. We then have $x_k = x_0 + V_k y_k$, where the vector y_k minimizes

$$\min_{y\in\mathbb{R}^k} \|b - A(x_0 + V_k y)\|_2 = \min_{y\in\mathbb{R}^k} \|r_0 - AV_k y\|_2. \tag{1.7.30}$$

This is a standard linear least-squares problem that can be solved by a QR decomposition.

One can use the modified Gram-Schmidt orthogonalization to find an orthonormal basis of $\|(A, r_0, k)$. The algorithm is given as follows:

Choose x_0, set $r_0 = b - Ax_0, v_1 = r_0/\|r_0\|_2$.
For $i = 1, 2, \cdots, k-1$, do:
  Compute $v_{i+1} = \frac{Av_i - \sum_{j=1}^{i}((Av_i)^{\mathrm{T}} v_j)v_j}{\left\|Av_i - \sum_{j=1}^{i}((Av_i)^{\mathrm{T}} v_j)v_j\right\|_2}$,
endFor

This algorithm produces the columns of the matrix V_k which also form an orthonormal basis for $\|(A, r_0, k)$. Note that the algorithm breaks down when a division by zero occurs.

If the modified Gram-Schmidt process does not break down, we can use it to carry out the GMRES method in the following efficient way. Let $h_{ij} = (Av_j)^{\mathrm{T}} v_i$. By the modified Gram-Schmidt algorithm, we have a $(k+1) \times k$ matrix H_k which is upper Hessenberg, i.e., its entries satisfy $h_{ij} = 0$ if $i > j + 1$. This process produces a sequence of matrices $\{V_k\}$ with orthonormal columns such that $AV_k = V_{k+1}H_k$. Therefore, we have

$$\begin{aligned} r_k &= b - Ax_k = r_0 - A(x_k - x_0) \\ &= \beta V_{k+1}e_1 - AV_k y_k = V_{k+1}(\beta e_1 - H_k y_k), \end{aligned} \tag{1.7.31}$$

where e_1 is the first unit k-vector $(1, 0, \cdots, 0)^{\mathrm{T}}$, and y_k is the solution of

$$\min_{y\in\mathbb{R}^k} \|\beta e_1 - H_k y\|_2. \tag{1.7.32}$$

Hence, $x_k = x_0 + V_k y_k$. To find a minimizer for (1.7.32), we need to look at the

linear algebraic system $\bar{H}_k y = \beta e_1$, namely,

$$\begin{pmatrix} h_{11} & h_{21} & \cdots & h_{k1} \\ h_{12} & h_{22} & \cdots & h_{k2} \\ & h_{23} & \cdots & h_{k3} \\ & & & \vdots \\ & & & h_{kk} \\ & & & h_{k+1,k} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_k \end{pmatrix} = \begin{pmatrix} \beta \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}.$$

This problem can be solved by using rotation matrices to do Gauss-elimination for $\bar{H}_k$ (see e.g. [134]), which yields $\overline{H}_k{}^{(k)} y = \overline{g}_k$, where

$$\overline{H}_k{}^{(k)} = \begin{pmatrix} h_{11}^{(k)} & h_{12}^{(k)} & \cdots & h_{1k}^{(k)} \\ & h_{22}^{(k)} & \cdots & h_{2k}^{(k)} \\ & & \ddots & \vdots \\ & & & h_{kk}^{(k)} \\ & & & h_{k+1,k} \end{pmatrix}, \quad \overline{g}_k = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_k \\ r_{k+1} \end{pmatrix}.$$

Moreover,

$$\min_{y \in \mathbb{R}^k} \|H_k y - \beta e_1\|_2 = \min_{y \in \mathbb{R}^k} \|\overline{H}_k{}^{(k)} y - \overline{g}_k\|_2. \tag{1.7.33}$$

Define $H_k^{(k)}$ to be the matrix containing the first m rows of $\overline{H}_k{}^{(k)}$. It is easy to see that the minimizer of (1.7.33) is the solution of $H_k^{(k)} y_k = \overline{g}_k$.

Below we give the GMRES algorithm for solving $Ax = b$ with A non-symmetric:

Choose x_0, set $r_0 = b - Ax_0$, $\beta = \|r_0\|_2$ and $v_1 = r_0/\beta$.
 For $j = 1, 2, \cdots, k, \cdots,$ do
  Compute $w_j = Av_j$
  for $i = 1, 2, \cdots, j$ do
   Compute $h_{ij} = w_j^{\mathrm{T}} v_i$.
   Set $w_j = w_j - h_{ij} v_i$.
  endfor
  Compute $h_{j+1,j} = \|w_j\|_2$
  Set $v_{j+1} = w_j / h_{j+1,j}$

endFor
Compute $H_k^{(k)}$ and $\overline{g}_k$
Solve $H_k^{(k)} y_k = \overline{g}_k$
Set $x_k = x_0 + V_k y_k$

Preconditioning techniques

It is seen from Theorem 1.7.1 that the rate of convergence of the conjugate gradient method depends on the condition number of A: the larger κ is, the closer γ will be to 1 and the slower will be the rate of convergence. A good preconditioner is a matrix M that is (i) easy to invert, and (ii) the condition number of $M^{-1}A$ is small, or the preconditioned system $M^{-1}Ax = M^{-1}b$ can be solved efficiently by an iterative method. This idea leads to the so-called **preconditioned conjugate gradient** (PCG) method:

Choose x_0, compute $r_0 = b - Ax_0$ and solve $M\tilde{r}_0 = r_0$
Set $p_0 = \tilde{r}_0$
For $k = 0, 1, \cdots$ do
 Compute $\alpha_k = -(\tilde{r}_k, r_k)/(p_k, Ap_k)$
 Set $x_{k+1} = x_k + \alpha_k p_k$; Set $r_{k+1} = r_k - \alpha_k A p_k$
 If $\|r_{k+1}\|_2 \geqslant \epsilon$, continue,
 Solve $M\tilde{r}_{k+1} = r_{k+1}$
 Compute $\beta_k = (\tilde{r}_{k+1}, r_{k+1})/(\tilde{r}_k, r_k)$
 Set $p_{k+1} = \tilde{r}_{k+1} + \beta_k p_k$
endFor

In the above algorithm, we need to solve the system $M\tilde{r} = r$ which may be as complicated as the original system. The idea for reducing the condition number of $M^{-1}A$ is to choose M such that M^{-1} is close to A^{-1}, while the system $M\tilde{r} = r$ is easy to solve. The following theorem describes a way to choose M.

Theorem 1.7.2 *Let A be an $n \times n$ nonsingular matrix and $A = P - Q$ a splitting of A such that P is nonsingular. If $H = P^{-1}Q$ and $\rho(H) < 1$, then*

$$A^{-1} = \left(\sum_{k=0}^{\infty} H^k \right) P^{-1}. \qquad \square$$

Based on this theorem, we can consider the matrices

$$M = P(I + H + \cdots + H^{m-1})^{-1},$$
$$M^{-1} = (I + H + \cdots + H^{m-1})P^{-1}$$

to be approximations of A and A^{-1}, respectively. Thus the solution of the system $M\tilde{r} = r$ becomes

$$\tilde{r} = M^{-1}r = (I + H + \cdots + H^{m-1})P^{-1}r.$$

Equivalently, the solution $\tilde{r} = r_m$ is the the result of applying m steps of the iterative method

$$Pr_{i+1} = Qr_i + r, \;\; i = 0, 1, \cdots, m-1, \;\; r_0 = 0.$$

If $P = D$, $Q = L + U$, the above iteration is the standard Jacobi iteration. Then in the PCG method we replace the system $M\tilde{r}_{k+1} = r_{k+1}$ with `do` m `Jacobi iterations on` $Ar = r_{k+1}$ `to obtain` $\tilde{r}_{k+1}$. The resulting method is called the m-**step Jacobi PCG Method**.

In practice, we may just use the one-step Jacobi PCG Method: in this case $M = D$. Similarly, the symmetric Gauss-Seidel and symmetric successive over-relaxation (SSOR) methods can also be used as preconditioners:

- Symmetric Gauss-Seidel preconditioner:

$$M = (D - L)D^{-1}(D - U), \quad M^{-1} = (D - U)^{-1}D(D - L)^{-1};$$

- SSOR preconditioner:

$$M = \frac{\omega}{2 - \omega}(\omega^{-1}D - L)D^{-1}(\omega^{-1}D - U),$$
$$M^{-1} = (2 - \omega)\omega(D - \omega U)^{-1}D(D - \omega L)^{-1}.$$

Preconditioned GMRES

If we use M as a left preconditioner for the GMRES method, then we are trying to minimize the residual in the space:

$$K_m(A, r_0) = \text{span}\{r_0, M^{-1}Ar_0, \cdots (M^{-1}A)^{m-1}r_0\}. \tag{1.7.34}$$

The resulting algorithm is exactly the same as the original GMRES, except that the matrix A is replaced by $M^{-1}A$.

Below is the preconditioned version of the **GMRES method with left-**

preconditioning:

Compute $r_0 = M^{-1}(b - Ax_0)$ and set $\beta = \|r_0\|_2,\ v_1 = r_0/\beta$.
For $j = 1, 2, \cdots, k, \cdots$ do:
  Compute $w_j = M^{-1}Av_j$.
  for $i = 1, 2, \cdots, j,$ do:
    Compute $h_{ij} = (w_j, v_i)$; Set $w_j = w_j - h_{ij}v_i$
  endfor
  Compute $h_{j+1,j} = \|w_j\|$.
  Set $v_{j+1} = w_j/h_{j+1,j}$.
endFor
Compute $H_k^{(k)}$ and $\overline{g}_k$
Solve $H_k^{(k)} y_k = \overline{g}_k$
Set $x_k = x_0 + V_k y_k$

If M is used as a right preconditioner, we just need to replace A in the original GMRES by AM^{-1}. Also, in the last step, we need to update x_k by

$$x_k = x_0 + M^{-1}V_k y_k. \tag{1.7.35}$$

In practice, for the GMRES method, however, the Gauss-Seidel and SOR methods can also be used as preconditioners:

- Gauss-Seidel preconditioner: $M = D - L,\ M^{-1} = (D - L)^{-1}$;
- SOR preconditioner: $M = \omega^{-1}D - L,\ M^{-1} = \omega(D - \omega L)^{-1}$.

The preconditioned CGS or BiCGSTAB algorithms can be constructed similarly. In general, to use preconditioners for the CGS or BiCGSTAB, we just need to replace the matrix A in the original algorithms by $M^{-1}A$ or AM^{-1}.

Exercise 1.7

Problem 1 Prove (1.7.5) and (1.7.7).

Problem 2 Prove Theorem 1.7.2.

1.8 Error estimates of polynomial approximations

Orthogonal projection in $L^2_{\omega^{\alpha,\beta}}(I)$
Orthogonal projection in $H^1_{0,\omega^{\alpha,\beta}}(I)$
Interpolation error

The numerical analysis of spectral approximations relies on the polynomial approximation results in various norms. In this section, we present some of the basic approximation results for the Jacobi polynomials which include the Legendre and Chebyshev polynomials as special cases. Some basic properties of the Jacobi polynomials are introduced in Section 1.4.

We first introduce some notations. Let $I = (-1, 1)$ and $\omega(x) > 0$ be a weight function (ω is not necessarily in $L^1(I)$). We define the "usual" weighted Sobolev spaces:

$$\begin{aligned} L^2_\omega(I) &= \left\{ u : \int_I u^2 \omega \mathrm{d}x < +\infty \right\}, \\ H^l_\omega(I) &= \left\{ u \in L^2_\omega(I) : \partial_x u, \cdots, \partial^l_x u \in L^2_\omega(I) \right\}, \\ H^l_{0,\omega}(I) &= \left\{ u \in H^l_\omega(I) : u(\pm 1) = \partial_x u(\pm 1) = \cdots = \partial^{l-1}_x u(\pm 1) = 0 \right\}. \end{aligned} \tag{1.8.1}$$

The norms in $L^2_\omega(I)$ and $H^l_\omega(I)$ will be denoted by $\|\cdot\|_\omega$ and $\|\cdot\|_{l,\omega}$, respectively. Furthermore, we shall use $|u|_{l,\omega} = \|\partial^l_x u\|_\omega$ to denote the semi-norm in $H^l_\omega(I)$. When $\omega(x) \equiv 1$, the subscript ω will often be omitted from the notations. Hereafter, we denote the Jacobi weight function of index (α, β) by

$$\omega^{\alpha,\beta}(x) = (1-x)^\alpha (1+x)^\beta.$$

It turns out that the "uniformly" weighted Sobolev spaces in (1.8.1) are not the most appropriate ones to describe the approximation error. Hence, we introduce the following non-uniformly weighted Sobolev spaces:

$$H^m_{\omega^{\alpha,\beta},*}(I) := \left\{ u : \partial^k_x u \in L^2_{\omega^{\alpha+k,\beta+k}}(I), \ \ 0 \leqslant k \leqslant m \right\}, \tag{1.8.2}$$

equipped with the inner product and norm

$$\big(u, v\big)_{m,\omega^{\alpha,\beta},*} = \sum_{k=0}^{m} (\partial^k_x u, \partial^k_x v)_{\omega^{\alpha+k,\beta+k}}, \quad \|u\|_{m,\omega^{\alpha,\beta},*} = \big(u, u\big)^{\frac{1}{2}}_{m,\omega^{\alpha,\beta},*}. \tag{1.8.3}$$

Hereafter, we shall use the expression $A_N \lesssim B_N$ to mean that there exists a positive constant C, independent of N, such that $A_N \leqslant CB_N$.

Orthogonal projection in $L^2_{\omega^{\alpha,\beta}}(I)$

Since $\{J_n^{\alpha,\beta}\}$ forms a complete orthogonal system in $L^2_{\omega^{\alpha,\beta}}(I)$, we can write

$$u(x) = \sum_{n=0}^{\infty} \hat{u}_n^{\alpha,\beta} J_n^{\alpha,\beta}(x), \quad \text{with} \quad \hat{u}_n^{\alpha,\beta} = \frac{(u, J_n^{\alpha,\beta})_{\omega^{\alpha,\beta}}}{\gamma_n^{\alpha,\beta}}, \tag{1.8.4}$$

where $\gamma_n^{\alpha,\beta} = \|J_n^{\alpha,\beta}\|^2_{\omega^{\alpha,\beta}}$. It is clear that

$$P_N = \text{span}\left\{J_0^{\alpha,\beta},\ J_1^{\alpha,\beta},\ \cdots,\ J_N^{\alpha,\beta}\right\}. \tag{1.8.5}$$

We start by establishing some fundamental approximation results on the $L^2_{\omega^{\alpha,\beta}}-$ orthogonal projection $\pi_{N,\omega^{\alpha,\beta}}:\ L^2_{\omega^{\alpha,\beta}}(I) \to P_N$, defined by

$$(\pi_{N,\omega^{\alpha,\beta}} u - u,\ v)_{\omega^{\alpha,\beta}} = 0, \quad \forall v \in P_N. \tag{1.8.6}$$

It is clear that $\pi_{N,\omega^{\alpha,\beta}} u$ is the best $L^2_{\omega^{\alpha,\beta}}-$approximate polynomial of u, and can be expressed as

$$(\pi_{N,\omega^{\alpha,\beta}} u)(x) = \sum_{n=0}^{N} \hat{u}_n^{\alpha,\beta} J_n^{\alpha,\beta}(x). \tag{1.8.7}$$

First of all, we derive inductively from (1.4.7) that

$$\partial_x^k J_n^{\alpha,\beta}(x) = d_{n,k}^{\alpha,\beta} J_{n-k}^{\alpha+k,\beta+k}(x), \quad n \geqslant k, \tag{1.8.8}$$

where

$$d_{n,k}^{\alpha,\beta} = \frac{\Gamma(n+k+\alpha+\beta+1)}{2^k \Gamma(n+\alpha+\beta+1)}. \tag{1.8.9}$$

As an immediate consequence of this formula and the orthogonality (1.4.5), we have

$$\int_{-1}^{1} \partial_x^k J_n^{\alpha,\beta}(x) \partial_x^k J_l^{\alpha,\beta}(x) \omega^{\alpha+k,\beta+k}(x) \mathrm{d}x = h_{n,k}^{\alpha,\beta} \delta_{n,l}, \tag{1.8.10}$$

where

$$h_{n,k}^{\alpha,\beta} = (d_{n,k}^{\alpha,\beta})^2 \gamma_{n-k}^{\alpha+k,\beta+k}. \tag{1.8.11}$$

Let us recall first Stirling's formula,

$$\Gamma(x)=\sqrt{2\pi}x^{x-1/2}e^{-x}\Big\{1+\frac{1}{12x}+\frac{1}{288x^2}+\mathcal{O}(x^{-3})\Big\}. \tag{1.8.12}$$

In particular, we have

$$\Gamma(n+1)=n!\cong\sqrt{2\pi}n^{n+1/2}e^{-n}, \tag{1.8.13}$$

which can be used to obtain the following asymptotic behaviors for $n\gg 1$:

$$\gamma_n^{\alpha,\beta}\sim n^{-1},\quad d_{n,k}^{\alpha,\beta}\sim n^k,\quad h_{n,k}^{\alpha,\beta}\sim n^{2k-1}. \tag{1.8.14}$$

Here, we have adopted the conventional assumption that α,β and k are small constants when compared with large n.

Below is the main result on the Jacobi projection error:

Theorem 1.8.1 *Let $\alpha,\beta>-1$. For any $u\in H^m_{\omega^{\alpha,\beta},*}(I)$ and $m\in\mathbb{N}$,*

$$\|\partial_x^l(\pi_{N,\omega^{\alpha,\beta}}u-u)\|_{\omega^{\alpha+l,\beta+l}}\lesssim N^{l-m}\|\partial_x^m u\|_{\omega^{\alpha+m,\beta+m}},\ \ 0\leqslant l\leqslant m. \tag{1.8.15}$$

Proof Owing to (1.8.10)∼(1.8.11), we have

$$\|\partial_x^k u\|^2_{\omega^{\alpha+k,\beta+k}}=\sum_{n=k}^{\infty}\big(\hat{u}_n^{\alpha,\beta}\big)^2\|\partial_x^k J_n^{\alpha,\beta}\|^2_{\omega^{\alpha+k,\beta+k}}, \tag{1.8.16}$$

$$\begin{aligned}\|\partial_x^l(\pi_{N,\omega^{\alpha,\beta}}u-u)\|^2_{\omega^{\alpha+l,\beta+l}}&=\sum_{n=N+1}^{\infty}\big(\hat{u}_n^{\alpha,\beta}\big)^2\|\partial_x^l J_n^{\alpha,\beta}\|^2_{\omega^{\alpha+l,\beta+l}}\\&=\sum_{n=N+1}^{\infty}\frac{h_{n,l}^{\alpha,\beta}}{h_{n,m}^{\alpha,\beta}}\big(\hat{u}_n^{\alpha,\beta}\big)^2\|\partial_x^m J_n^{\alpha,\beta}\|^2_{\omega^{\alpha+m,\beta+m}}.\end{aligned} \tag{1.8.17}$$

Using the the asymptotic estimate (1.8.14) gives

$$h_{n,l}^{\alpha,\beta}/h_{n,m}^{\alpha,\beta}\lesssim n^{2(l-m)},\quad n\gg 1,\ l,m\in\mathbb{N},$$

which, together with (1.8.17), leads to

$$\begin{aligned}\|\partial_x^l(\pi_{N,\omega^{\alpha,\beta}}u-u)\|^2_{\omega^{\alpha+l,\beta+l}}&\lesssim(N+1)^{2(l-m)}\sum_{n=N+1}^{\infty}\big(\hat{u}_n^{\alpha,\beta}\big)^2\|\partial_x^m J_n^{\alpha,\beta}\|^2_{\omega^{\alpha+m,\beta+m}}\\&\lesssim N^{2(l-m)}\|\partial_x^m u\|^2_{\omega^{\alpha+m,\beta+m}}.\end{aligned}$$

This ends the proof. □

We shall now extend the above result to the cases where α and/or β are negative integers, using the properties of the generalized Jacobi polynomials. We point out that like the classical Jacobi polynomials, the GJPs with negative integer indexes form a complete orthogonal system in $L^2_{\omega^{k,l}}(I)$.

Hence, we define the polynomial space

$$Q_N^{k,l} := \text{span}\{J_{n_0}^{k,l}, J_{n_0+1}^{k,l}, \cdots, J_N^{k,l}\}, \quad k \leqslant -1 \ \text{ and/or } \ l \leqslant -1, \tag{1.8.18}$$

where n_0 is defined in (1.4.8). According to Remark 1.4.1, we have that for $k < -1$ and/or $l \leqslant -1$,

$$Q_N^{k,l} = \{\phi \in P_N \ : \ \partial_x^i \phi(-1) = \partial_x^j \phi(1) = 0, \ \ 0 \leqslant i \leqslant -k-1, \ 0 \leqslant j \leqslant -l-1\}.$$

We now define the orthogonal projection $\pi_{N,\omega^{k,l}} : \ L^2_{\omega^{k,l}}(I) \to Q_N^{k,l}$ by

$$(u - \pi_{N,\omega^{k,l}} u, v_N)_{\omega^{k,l}} = 0, \quad \forall v_N \in Q_N^{k,l}. \tag{1.8.19}$$

Owing to the orthogonality (1.4.10) and the derivative relation (1.4.13), the following theorem is a direct extension of Theorem 1.8.1.

Theorem 1.8.2 *For any $k, l \in \mathbb{Z}$, and $u \in H^m_{\omega^{k,l},*}(I)$,*

$$\|\partial_x^\mu(\pi_{N,\omega^{k,l}} u - u)\|_{\omega^{k+\mu,l+\mu}} \lesssim N^{\mu-m} \|\partial_x^m u\|_{\omega^{k+m,l+m}}, \ \ 0 \leqslant \mu \leqslant m. \tag{1.8.20}$$

Orthogonal projection in $H^1_{0,\omega^{\alpha,\beta}}(I)$

In order to carry out the error analysis of spectral methods for second-order elliptic equations with Dirichlet boundary conditions, we need to study the orthogonal projection error in the space $H^1_{0,\omega^{\alpha,\beta}}(I)$. We define

$$P_N^0 = \{u \in P_N : \ u(\pm 1) = 0\}. \tag{1.8.21}$$

Definition 1.8.1 The orthogonal projector $\pi^{1,0}_{N,\omega^{\alpha,\beta}} : \ H^1_{0,\omega^{\alpha,\beta}}(I) \to P_N^0$ is defined by

$$((u - \pi^{1,0}_{N,\omega^{\alpha,\beta}} u)', v')_{\omega^{\alpha,\beta}} = 0, \quad \forall \ v \in P_N^0. \tag{1.8.22}$$

Theorem 1.8.3 *Let $-1 < \alpha, \beta < 1$. Then for any $u \in H^1_{0,\omega^{\alpha,\beta}}(I) \cap H^m_{\omega^{\alpha-1,\beta-1},*}(I)$,*

$$\|\partial_x(u - \pi^{1,0}_{N,\omega^{\alpha,\beta}} u)\|_{\omega^{\alpha,\beta}} \lesssim N^{1-m} \|\partial_x^m u\|_{\omega^{\alpha+m-1,\beta+m-1}}, \qquad m \geqslant 1.$$

Proof For any $u \in H^1_{0,\omega^{\alpha,\beta}}(I)$, we set

$$u_N = \int_{-1}^{x} \Big\{ \pi_{N-1,\omega^{\alpha,\beta}} u' - \frac{1}{2} \int_{-1}^{1} \pi_{N-1,\omega^{\alpha,\beta}} u' \mathrm{d}\eta \Big\} \mathrm{d}\xi. \tag{1.8.23}$$

Therefore,

$$u_N \in P_N^0 \ \text{ and } u'_N = \pi_{N-1,\omega^{\alpha,\beta}} u' - \frac{1}{2} \int_{-1}^{1} \pi_{N-1,\omega^{\alpha,\beta}} u' \mathrm{d}\eta.$$

Hence,

$$\|u' - u'_N\|_{L^2_{\omega^{\alpha,\beta}}} \leqslant \|u' - \pi_{N-1,\omega^{\alpha,\beta}} u'\|_{L^2_{\omega^{\alpha,\beta}}} + \Big| \frac{1}{2} \int_{-1}^{1} \pi_{N-1,\omega^{\alpha,\beta}} u' \mathrm{d}\eta \Big|. \tag{1.8.24}$$

On the other hand, since $u(\pm 1) = 0$, we derive by using the Cauchy-Schwarz inequality that

$$\begin{aligned} &\Big| \int_{-1}^{1} \pi_{N-1,\omega^{\alpha,\beta}} u' \mathrm{d}x \Big| = \Big| \int_{-1}^{1} (\pi_{N-1,\omega^{\alpha,\beta}} u' - u') \mathrm{d}x \Big| \\ \leqslant &\Big(\int_{-1}^{1} (\omega^{\alpha,\beta})^{-1} \mathrm{d}x \Big)^{\frac{1}{2}} \|\pi_{N-1,\omega^{\alpha,\beta}} u' - u'\|_{L^2_{\omega^{\alpha,\beta}}} \lesssim \|\pi_{N-1,\omega^{\alpha,\beta}} u' - u'\|_{L^2_{\omega^{\alpha,\beta}}}, \end{aligned} \tag{1.8.25}$$

for $\alpha, \beta < 1$. We then conclude from (1.8.24), (1.8.25) and Theorem 1.8.1 that

$$\begin{aligned} \|\partial_x (u - \pi^{1,0}_{N,\omega^{\alpha,\beta}} u)\|_{\omega^{\alpha,\beta}} &= \inf_{\phi_N \in P_N^0} \|u' - \phi'_N\|_{\omega^{\alpha,\beta}} \leqslant \|u' - u'_N\|_{\omega^{\alpha,\beta}} \\ &\lesssim \|u' - \pi_{N-1,\omega^{\alpha,\beta}} u'\|_{\omega^{\alpha,\beta}} \lesssim N^{1-m} \|\partial_x^m u\|_{\omega^{\alpha+m-1,\beta+m-1}}. \end{aligned}$$

This completes the proof of Theorem 1.8.3. □

Interpolation error

We present below an optimal error estimate for the interpolation polynomials based on the Gauss-Lobatto points.

Theorem 1.8.4 *Let $\{x_j\}_{j=0}^N$ be the roots of $(1-x^2)\partial_x J_N^{\alpha,\beta}(x)$ with $-1 < \alpha, \beta < 1$. Let $I_{N,\omega^{\alpha,\beta}} : C[-1,1] \to P_N$ be the interpolation operator with respect to $\{x_j\}_{j=0}^N$. Then, we have*

$$\|\partial_x^l (I_N^{\alpha,\beta} u - u)\|_{\omega^{\alpha+l,\beta+l}} \lesssim N^{l-m} \|\partial_x^m u\|_{\omega^{\alpha+m,\beta+m}}, \ \ 0 \leqslant l \leqslant m. \tag{1.8.26}$$

The proof of the above lemma is rather technical. We refer to [3] for a complete proof (see also [11] for a similar result for the special case $\alpha = \beta$).

Theorem 1.8.4 indicates that error estimates for the interpolation polynomial based on the Gauss-Lobatto points are optimal in suitable weighted Sobolev spaces. One should note that an interpolation polynomial based on uniformly spaced points is usually a very poor approximation unless the function is periodic in the concerned interval.

As we can see from the estimates presented in this section, the convergence rates of spectral projection/interpolation increase with the smoothness of the function, as opposed to a fixed convergence rate for the finite difference or finite element approximations. Moreover, it can be shown that the convergence rates of spectral projection/interpolation are exponential for analytical functions. We now provide a direct proof of this statement in the Chebyshev case.

Let $\{x_j\}$ be the set of Chebyshev-Gauss-Lobatto points, i.e. $x_0 = 1$, $x_N = -1$ and $T'_N(x_j) = 0, 1 \leqslant j \leqslant N-1$. This suggests that

$$T'_N(x) = \alpha_N \prod_{j=1}^{N-1} (x - x_j).$$

Since $T_N(x) = 2^{N-1}\hat{T}_N(x)$, where $\hat{T}_N(x)$ is monic, we have

$$T_N(x) = 2^{N-1}x^N + \text{lower order terms}.$$

Combining the above two equations gives $\alpha_N = N2^{N-1}$. Notice also that $x_0 = 1$ and $x_N = -1$, we obtain

$$\prod_{k=0}^{N} (x - x_k) = \frac{2^{1-N}}{N} (x^2 - 1) T'_N(x).$$

The above result, together with (1.3.6a), yields

$$\Big| \prod_{k=0}^{N} (x - x_k) \Big| \leqslant N2^{1-N}. \tag{1.8.27}$$

Let u be a smooth function in $C^{N+1}(-1, 1)$. Using Lemma 1.2.3, (1.8.27) and Stir-

ling's formula (1.8.13), we obtain

$$\max_{x\in\bar{I}} |u(x) - I_{N,\omega^{\alpha,\beta}} u(x)| \leqslant C\|u^{(N+1)}\|_\infty \left(\frac{e}{2N}\right)^N, \tag{1.8.28}$$

for large N, where C is a constant independent of N. This result implies that if u is smooth, then the interpolations using the Chebyshev-Gauss-Lobatto points may lead to *exponential order* of convergence.

Exercise 1.8

Problem 1 Prove Theorem 1.8.2.

Problem 2 Show that $\pi_{N,\omega^{-1,-1}} = \pi^{1,0}_{N,\omega^{0,0}}$.

Chapter 2

Spectral-Collocation Methods

Contents

2.1	**Differentiation matrices for polynomial basis functions . . .**	**69**
2.2	**Differentiation matrices for Fourier collocation methods . .**	**79**
2.3	**Eigenvalues of Chebyshev collocation operators . . . . . .**	**84**
2.4	**Chebyshev collocation method for two-point BVPs . . . . . .**	**91**
2.5	**Collocation method in the weak form and preconditioning .**	**99**

The collocation method① is the most popular form of the spectral methods among practitioners. It is very easy to implement, in particular for one-dimensional problems, even for very complicated nonlinear equations, and generally leads to satisfactory results as long as the problems possess sufficient smoothness.

We present in this chapter some basic ingredients for the spectral collocation methods. In the first two sections, we describe how to compute the differentiation matrices associated with the Chebyshev and Fourier collocation. In Section 2.4, we present in detail a Chebyshev collocation method for two-point boundary value problems with general boundary conditions. We study in Section 2.3 the spectral radius and condition number of the Chebyshev collocation approximation to the advection

① In the literature on spectral methods, the terms collocation and pseudospectral (PS) are often used in a interchangeable fashion. Strictly speaking, a collocation method seeks an approximate solution to satisfy the underlying equation at a set of collocation points, while a method is pseudospectral if not all parts of the algorithm are performed in a pure spectral fashion. Therefore, a collocation method is always a pseudospectral method while a psudospectral method is not necessarily a collocation method.

and diffusion operators. In Section 2.5, we present a weak formulation of the collocation method and discuss how to construct effective preconditioners for spectral-collocation methods.

2.1 Differentiation matrices for polynomial basis functions

Polynomial basis functions
Finite-difference weights on arbitrary grids
Differentiation matrices using recursive formulas
Differentiation matrices using direct formulas

Differentiation matrices play an important role in the implementation of spectral collocation method. In order to introduce the differentiation matrix idea, let us consider, as an example, the differentiation matrix associated with the finite difference method for the model problem

$$u_{xx} = f, \quad x \in (-1, 1); \qquad u(\pm 1) = 0. \tag{2.1.1}$$

Let us denote $x_j = -1 + jh, 0 \leqslant j \leqslant N$, with $h = 2/N$. A finite difference method for (2.1.1) is to approximate u_{xx} by the central difference formula:

$$u_{xx}(x) \approx \frac{1}{h^2}[u(x+h) - 2u(x) + u(x-h)].$$

Since the solutions of the continuous problem (2.1.1) and the discrete problem are different, we use U to denote the solution of the discrete problem. One can easily verify that the discrete solution satisfies

$$\begin{pmatrix} -2/h^2 & 1/h^2 & 0 & \cdots & 0 \\ 1/h^2 & -2/h^2 & 1/h^2 & \cdots & 0 \\ 0 & 1/h^2 & -2/h^2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & -2/h^2 \end{pmatrix} \begin{pmatrix} U(x_1) \\ U(x_2) \\ U(x_3) \\ \vdots \\ U(x_{N-1}) \end{pmatrix} = \begin{pmatrix} f(x_1) \\ f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{N-1}) \end{pmatrix}. \tag{2.1.2}$$

The matrix above is the so-called *differentiation matrix* (DM) of the finite-difference method for the second-order derivative. In general, for a problem involving the m-th

derivative $u^{(m)}$, the differentiation matrix is defined by

$$D^m = \left(d_{ij}^{(m)}\right)_{i,j=0}^{N};$$

it satisfies

$$\begin{pmatrix} u^{(m)}(x_0) \\ u^{(m)}(x_1) \\ \vdots \\ u^{(m)}(x_N) \end{pmatrix} = D^m \begin{pmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_N) \end{pmatrix}.$$

In this section, we will discuss how to find the DM for the spectral collocation methods when the basis functions are polynomial (e.g. the Chebyshev polynomials, Legendre polynomials, Hermite polynomials etc). The DM is dependent on the collocation points and the chosen basis functions.

Polynomial basis functions

If the basis functions $\Phi_k(x)$ are polynomials, the spectral approximation is of the form $u^N(x) = \sum_{k=0}^{N} a_k \Phi_k(x)$, where the coefficients a_k can be determined from a given set of collocation points $\{x_j\}_{j=0}^{N}$ and the function values $u^N(x_j)$. Since $u^N(x)$ is a polynomial, it can also be written in the form

$$u^N(x) = \sum_{k=0}^{N} u^N(x_k) F_k(x), \tag{2.1.3}$$

where the $F_k(x)$ are called *Lagrange polynomials* which satisfy

$$F_k(x_j) = \begin{cases} 0 & \text{if} \quad k \neq j, \\ 1 & \text{if} \quad k = j. \end{cases}$$

We will use (2.1.3), the equivalent form of $u^N(x)$, to obtain the differentiation matrices for polynomial basis functions. If the basis functions are not polynomials (e.g. trigonometric functions), the equivalent form does not exist and the codes given in this section will not work.

Finite-difference weights on arbitrary grids

We now describe a simple recursive relation which gives the weights for any order of derivative, approximated to any order of accuracy on an arbitrary grid in one dimension. This simple recursive relation was introduced in [46].

Given $M \geqslant 0$, the order of the highest derivative we wish to approximate, and a set of $N+1$ grid points (at x-coordinates $\alpha_0, \cdots, \alpha_N$; $N \geqslant 0$), the problem is to find all the weights such that the approximations

$$\frac{d^m f}{dx^m}\Big|_{x=\zeta} \approx \sum_{\nu=0}^{n} c_{n,\nu}^m(\zeta) f(\alpha_\nu), \quad m = 0, 1, \cdots, M;\ n = m, m+1, \cdots, N,$$

possess a (formal) optimal order of accuracy (in general of order $n-m+1$, although it can be higher in special cases).

For simplicity, assume we seek to approximate the derivatives at the point $\zeta = 0$ (for a nonzero point ζ a simple shift will work). Let $\{\alpha_0, \alpha_1, \cdots, \alpha_N\}$ be distinct real numbers and define

$$\gamma_n(x) := \prod_{k=0}^{n} (x - \alpha_k).$$

The polynomial

$$F_{n,\nu}(x) := \frac{\gamma_n(x)}{\gamma_n'(\alpha_\nu)(x - \alpha_\nu)} \tag{2.1.4}$$

is the one of minimal degree which takes the value 1 at $x = \alpha_\nu$ and 0 at $x = \alpha_k, k \neq \nu$. For an arbitrary function $f(x)$ and nodes $x = \alpha_\nu$, Lagrange's interpolation polynomial becomes

$$p(x) := \sum_{\nu=0}^{n} F_{n,\nu}(x) f(\alpha_\nu).$$

The desired weights express how the values of $[d^m p(x)/dx^m]_{x=0}$ vary with changes in $f(\alpha_\nu)$. Since only one term in $p(x)$ is influenced by changes in each $f(\alpha_\nu)$, we find

$$c_{n,\nu}^m = \left[\frac{d^m}{dx^m} F_{n,\nu}(x)\right]_{x=0}. \tag{2.1.5}$$

Therefore, the n-th degree polynomial $F_{n,\nu}(x)$ can also be expressed as

$$F_{n,\nu}(x) = \sum_{m=0}^{n} \frac{c_{n,\nu}^m}{m!} x^m. \tag{2.1.6}$$

Noting that $\gamma_n(x) = (x-\alpha_n)\gamma_{n-1}(x)$ implies $\gamma_n'(x) = (x-\alpha_n)\gamma_{n-1}'(x) + \gamma_{n-1}(x)$.

It follows from (2.1.4) that

$$\begin{aligned} F_{n,\nu}(x) &= \frac{x-\alpha_n}{\alpha_\nu-\alpha_n}F_{n-1,\nu}(x), \quad \text{for} \quad \nu<n; \\ F_{n,n}(x) &= \frac{\gamma_{n-1}(x)}{\gamma_{n-1}(\alpha_n)} = \frac{\gamma_{n-2}(\alpha_{n-1})}{\gamma_{n-1}(\alpha_n)}(x-\alpha_{n-1})F_{n-1,n-1}(x) \quad (n>1). \end{aligned} \tag{2.1.7}$$

By substituting the expression (2.1.6) into the above two equations, and by equating powers of x, the desired recursive relations for the weights are obtained:

$$c_{n,\nu}^m = \frac{1}{\alpha_n-\alpha_\nu}\left(\alpha_n c_{n-1,\nu}^m - m c_{n-1,\nu}^{m-1}\right) \quad \text{for} \quad \nu<n \tag{2.1.8a}$$

$$c_{n,n}^m = \frac{\gamma_{n-2}(\alpha_{n-1})}{\gamma_{n-1}(\alpha_n)}\left(m c_{n-1,n-1}^{m-1} - \alpha_{n-1} c_{n-1,n-1}^m\right). \tag{2.1.8b}$$

The relation

$$\sum_{\nu=0}^{n} c_{n,\nu}^m = 0 \quad \text{for} \quad m>0; \qquad \sum_{\nu=0}^{n} c_{n,\nu}^0 = 1, \tag{2.1.9}$$

can be used instead of (2.1.8b) to obtain $c_{n,n}^m$. However, this would increase the operation count and might also cause a growth of errors in the case of floating arithmetic.

It is obvious that $c_{0,0}^0 = 1$. Using this fact, together with (2.1.8a), we obtain

$$c_{1,0}^0,\ c_{1,0}^1,\ \cdots,\ c_{1,0}^M.$$

Then, using $c_{0,0}^0 = 1$ and (2.1.8b) leads to

$$c_{1,1}^0,\ c_{1,1}^1,\ \cdots,\ c_{1,1}^M.$$

The above information, together with (2.1.8a), give

$$\begin{aligned} &c_{2,0}^0,\ c_{2,0}^1,\ \cdots,\ c_{2,0}^M; \\ &c_{2,1}^0,\ c_{2,1}^1,\ \cdots,\ c_{2,1}^M. \end{aligned}$$

Using (2.1.8b) or (2.1.9), we can find

$$c_{2,2}^0,\ c_{2,2}^1,\ \cdots,\ c_{2,2}^M.$$

Repeating the above process will generate all the coefficients $c_{n,\nu}^m$, for $m \leqslant n \leqslant N, 0 \leqslant \nu \leqslant n$.

In practice, we wish to use all of the information $f(\alpha_\nu), 0 \leqslant \nu \leqslant N$. Therefore,

it is of interest to compute $c_{N,\nu}^M$, $0 \leqslant \nu \leqslant N$, for given values of M and N. The following pseudocode is designed for this purpose. We let $\alpha = (\alpha_0, \alpha_1, \cdots, \alpha_N)^{\mathrm{T}}$.

```
CODE DM.1
Function d=FDMx(M,N,ζ,α)
 c^0_{0,0}=1, c_1=1
 for n=1 to N do
    c_2=1
    for ν=0 to n-1 do
        c_3=α_n-α_ν,   c_2=c_2*c_3
        for m=0 to M do
            c^m_{n,ν}=( (α_n-ζ)*c^m_{n-1,ν}-m*c^{m-1}_{n-1,ν} )/c_3
        endfor
    endfor
    for m=0 to M do
        c^m_{n,n}=c_1 ( m c^{m-1}_{n-1,n-1} - (α_{n-1} - ζ) * c^m_{n-1,n-1} )/c_2
    endfor
 c_1=c_2
 endfor
 for j=0 to N do
    d(j)=c^M_{N,j}
 endfor
```

Differentiation matrices using recursive formulas

For non-periodic problems, the algorithm in the last section can be used to generate DMs very conveniently. Assume that the collocation points $x_j, 0 \leqslant j \leqslant N$ are provided. It is noted that

$$D^m = \left(c_{N,\nu}^m(x_j)\right)_{\nu,j=0}^N .$$

Let $\vec{x} = (x_0, x_1, \cdots, x_N)^{\mathrm{T}}$. A pseudocode to generate the matrix D^m is given below:

```
CODE DM.2
Input m, N, x⃗
   for j=0 to N do
      ζ=x_j
      d=FDMx(m,N,ζ,x⃗)
        for ν=0 to N do
           Dm(j, ν)=d(ν)
        endfor
   endfor
```

As an example, we compute D^1 and D^2 with $N = 4$ and $x_j = \cos(\pi j/N)$ (i.e.

the Chebyshev-Gauss-Lobatto points) by using `CODE DM.2`. The results are given below:

$$D^1 = \begin{pmatrix} 5.5000 & -6.8284 & 2.0000 & -1.1716 & 0.5000 \\ 1.7071 & -0.7071 & -1.4142 & 0.7071 & -0.2929 \\ -0.5000 & 1.4142 & 0.0000 & -1.4142 & 0.5000 \\ 0.2929 & -0.7071 & 1.4142 & 0.7071 & -1.7071 \\ -0.5000 & 1.1716 & -2.0000 & 6.8284 & -5.5000 \end{pmatrix},$$

$$D^2 = \begin{pmatrix} 17.0000 & -28.4853 & 18.0000 & -11.5147 & 5.0000 \\ 9.2426 & -14.0000 & 6.0000 & -2.0000 & 0.7574 \\ -1.0000 & 4.0000 & -6.0000 & 4.0000 & -1.0000 \\ 0.7574 & -2.0000 & 6.0000 & -14.0000 & 9.2426 \\ 5.0000 & -11.5147 & 18.0000 & -28.4853 & 17.0000 \end{pmatrix}. \tag{2.1.10}$$

It is observed from the above results that the following symmetry results hold:

$$D^1_{N-k,N-j} = -D^1_{kj}, \qquad D^2_{N-k,N-j} = D^2_{kj}, \qquad 0 \leqslant k, j \leqslant N.$$

In fact, this is true for any N if the collocation points are the Chebyshev-Gauss-Lobatto points (1.3.11).

Differentiation matrices using direct formulas

For some choices of collocation points, D^1 and D^2 can be found explicitly. To see this, we consider the Chebyshev points $x_j = \cos(\pi j/N), 0 \leqslant j \leqslant N$.

First we need the following results:

$$T'_N(x_j) = 0, \qquad 1 \leqslant j \leqslant N-1, \tag{2.1.11a}$$

$$T''_N(x_j) = (-1)^{j+1} N^2 \frac{1}{1-x_j^2}, \quad 1 \leqslant j \leqslant N-1, \tag{2.1.11b}$$

$$T'''_N(x_j) = (-1)^{j+1} 3N^2 \frac{x_j}{(1-x_j^2)^2}, \quad 1 \leqslant j \leqslant N-1, \tag{2.1.11c}$$

$$T'_N(\pm 1) = (\pm 1)^N N^2, \qquad T''_N(\pm 1) = \frac{1}{3}(\pm 1)^N N^2 (N^2-1). \tag{2.1.11d}$$

We briefly prove the above results. Let $\theta = \cos^{-1} x$. From the definition $T_N(x) = \cos(N\theta)$, we can show that

$$T'_N(x) = N \sin N\theta \frac{1}{\sin\theta}; \tag{2.1.12a}$$

$$T_N''(x) = -N^2 \cos N\theta \frac{1}{\sin^2\theta} + N \sin N\theta \frac{\cos\theta}{\sin^3\theta}; \tag{2.1.12b}$$

$$T_N'''(x) = N^3 \frac{\sin N\theta}{\sin^3\theta} - 3N^2 \cos N\theta \frac{\cos\theta}{\sin^4\theta} - N\frac{d}{d\theta}\left(\frac{\cos\theta}{\sin^3\theta}\right)\cdot\left(\frac{\sin N\theta}{\sin\theta}\right). \tag{2.1.12c}$$

Using these expressions and the fact that $\sin(N\theta_j) = 0$ ($\theta_j = \pi j/N$), we obtain (2.1.11a), (2.1.11b) and (2.1.11c). Letting $\theta \to 0$ and $\theta \to \pi$ in (2.1.12a), respectively, gives the first result in (2.1.11d). To obtain the second one, we use L'Hospital's rule for (2.1.12b):

$$\begin{aligned} T_N''(1) &= \lim_{\theta\to 0} \frac{1}{\sin^3\theta}\Big(-N^2 \cos N\theta \sin\theta + N \sin N\theta \cos\theta\Big) \\ &= \lim_{\theta\to 0} \frac{1}{3\sin\theta^2\cos\theta}\Big((N^3 - N)\sin N\theta \sin\theta\Big) = \frac{N^2}{3}(N^2-1). \end{aligned}$$

A similar procedure gives $T_N''(-1)$. Let $\gamma_N(x) = \prod_{k=0}^{N}(x - x_k)$. By (2.1.11a) we derive

$$\gamma_N(x) = \beta_N(x^2-1)T_N'(x),$$

where β_N is a positive constant such that the coefficient of the x^{N+1} term on the right-hand side of the above equality is 1. It follows from (2.1.11a) and (2.1.11d) that

$$\gamma_N'(x_j) = (-1)^j \tilde{c}_j N^2 \beta_N, \qquad 0 \leqslant j \leqslant N,$$

where $\tilde{c}_0 = \tilde{c}_N = 2$ and $\tilde{c}_j = 1$ for $1 \leqslant j \leqslant N-1$. Similar to (2.1.4), the Lagrange polynomials associated with $\{x_j\}_{j=0}^{N}$ are of the form

$$F_j(x) = \frac{\gamma_N(x)}{\gamma_N'(x_j)(x - x_j)}, \qquad 0 \leqslant j \leqslant N.$$

Using the expressions for $\gamma_N(x)$ and $\gamma_N(x_j)$ given above yields

$$F_j(x) = \frac{(-1)^j(x^2-1)T_N'(x)}{\tilde{c}_j N^2 (x - x_j)}, \qquad 0 \leqslant j \leqslant N. \tag{2.1.13}$$

Now direct calculation gives

$$F_j'(x) = \frac{(-1)^j}{\tilde{c}_j N^2}\frac{1}{(x-x_j)^2}\Big((2xT_N'(x) + (x^2-1)T_N''(x))(x-x_j) - (x^2-1)T_N'(x)\Big).$$

For $k \neq j$, the above result, together with (2.1.11a) and (2.1.11d), leads to

$$F_j'(x_k) = \frac{\tilde{c}_k}{\tilde{c}_j}\frac{(-1)^{k+j}}{x_k - x_j}, \qquad 0 \leqslant k \neq j \leqslant N.$$

For $1 \leqslant k = j \leqslant N-1$, it follows from (2.1.13) that

$$\begin{aligned} F_k'(x_k) &= \lim_{x\to x_k} \frac{F_k(x)-1}{x-x_k} \quad \Big(F_k(x_k) = 1\Big) \\ &= \lim_{x\to x_k} \frac{\alpha_{N,k}(x^2-1)T_N'(x) - (x-x_k)}{(x-x_k)^2}, \end{aligned}$$

where

$$\alpha_{N,k} := (-1)^k/N^2. \tag{2.1.14}$$

Again using L'Hospital's rule to the above result twice gives

$$\begin{aligned} F_k'(x_k) &= \frac{1}{2}\alpha_{N,k}\lim_{x\to x_k}\Big(2T_N'(x) + 4xT_N''(x) + (x^2-1)T_N'''(x)\Big) \\ &= \frac{1}{2}\alpha_{N,k}(-1)^{k+1}N^2\left(\frac{4x_k}{1-x_k^2} - \frac{3x_k}{1-x_k^2}\right) = -\frac{x_k}{2(1-x_k^2)}, \quad 1\leqslant k\leqslant N-1, \end{aligned}$$

where in the last step we have used (2.1.11a), (2.1.11b) and (2.1.11c). Further, using (2.1.11d) shows that

$$F_0'(x_0) = -F_N'(x_N) = (2N^2+1)/6.$$

Since the Lagrange's interpolation polynomial is of the form $p(x) = \sum_{j=0}^N F_j(x)$ $f(\alpha_j)$, we obtain by the definition of the differentiation matrix that

$$\big(D^1\big)_{kj} = F_j'(x_k).$$

The above discussion gives

$$D^1_{kj} = \frac{\tilde{c}_k}{\tilde{c}_j}\frac{(-1)^{k+j}}{x_k - x_j}, \qquad j \neq k \tag{2.1.15a}$$

$$D^1_{kk} = -\frac{x_k}{2(1-x_k^2)}, \qquad k \neq 0, N, \tag{2.1.15b}$$

$$D^1_{00} = -D^1_{NN} = (2N^2+1)/6, \tag{2.1.15c}$$

where $\tilde{c}_k = 1$, except for $\tilde{c}_0 = \tilde{c}_N = 2$. Direct verification from (2.1.15a) also yields

$$D^1_{N-k,N-j} = -D_{kj}. \tag{2.1.16}$$

It has been observed that for large N the direct implementation of the above formulas suffers from cancellation, causing errors in the elements of the matrix D^1. Thus, it is advisable to replace the first two formulas using trigonometric identities by the formulas

$$D^1_{kj} = \frac{\tilde{c}_k}{\tilde{c}_j}\frac{(-1)^{k+j}}{2}\left(\sin\frac{(j+k)\pi}{2N}\sin\frac{(j-k)\pi}{2N}\right)^{-1}, \qquad k \neq j, \tag{2.1.17a}$$

$$D^1_{kk} = -\frac{x_k}{2\sin^2(k\pi/N)}, \qquad k \neq 0, N. \tag{2.1.17b}$$

Finally, to avoid computing the sine of arguments larger than $\pi/2$ in absolute value we take advantage of the symmetry property (2.1.16). Thus the most accurate method of computing D^1 is using formulas (2.1.17) to find the *upper left triangle* of D^1 (i.e., compute D_{kj} with $k + j \leqslant N$), and then uses the relation (2.1.16) and (2.1.15c) for the other elements.

Higher-order DMs can be computed easily by the following observation:

> If the collocation points are the Chebyshev-Gauss-Lobatto points $x_j = \cos(\pi j/N)$, then higher derivative matrices can be obtained as matrix powers, i.e.,
>
> $$D^m = (D^1)^m. \tag{2.1.18}$$

The numerical results (2.1.10) obtained in the last subsection, i.e., D^1 and D^2 with $N = 4$, can be verified using the above explicit formulas.

A pseudocode for first order DM using the formula (2.1.15a)–(2.1.16) is given below:

```
CODE DM.3
Input N
Compute collocation points x(j)=cos(πj/N) and c̃(j)
%first order differentiation matrix
for k=0 to N do
   for j=0 to N-k do
      if k=0 and j=0 then D1(k,j)=(2N^2+1)/6
      elseif k=N and j=N then D1(k,j)=-D1(0,0)
      elseif k=j then D1(k,j)=-x(k)/(2*(1-x(k)^2))
      else then D1(k,j)=c̃(k)*(-1)^(j+k)/(c̃(j)*(x(k)-x(j)))
      endif
   endfor
```

```
endfor
for k=1 to N do
   for j=N-k+1 to N do
      D1(k,j)=-D1(N-k,N-j)
   endfor
endfor
```

Since we will use the first-order differentiation matrix frequently, it is also necessary to provide a MATLAB code for the above algorithm.

```
CODE DM.4
function d=DM1(N)
%collocation points, and c̃_k
j=[0:1:N];  x=[cos(pi*j/N)];
c=[2 ones(1,N-1) 2];
%Differentiation matrix
for k=1:N+1
   for j=1:N+2-k
      if j==1 & k==1
        d(j,k)=(2*N^2+1)/6;
      elseif j==N+1 & k==N+1
        d(j,k)=-d(1,1);
      elseif j==k
        d(j,k)=-x(k)/(2*(1-x(k)^2));
      else
        d(j,k)=c(j)*(-1)^(j+k)/(c(k)*(x(j)-x(k)));
      end
    end
 end
 for k=2:N+1
    for j=N+3-k:N+1
       d(k,j)=-d(N-k+2,N-j+2);
    end
 end
```

Remark 2.1.1 It is noted that $d(i,j) = D^1_{i-1,j-1}$ for $1 \leqslant i,j \leqslant N+1$ (since MATLAB requires that the indexes i and j above are positive).

Exercise 2.1

Problem 1 Consider the Legendre polynomial described in Section 1.3, with the Legendre-Gauss-Lobatto points (1.3.11). It can be verified that the Lagrange polynomials are of the form

$$F_j(x) = \frac{1}{N(N+1)L_N(x)} \frac{(1-x^2)L'_N(x)}{x - x_j}.$$

Use this result to verify that

$$D^1_{kj} = \frac{L_N(x_k)}{L_N(x_j)}\frac{1}{x_k - x_j}, \qquad k \neq j,$$
$$D^1_{kk} = 0, \qquad k \neq 0, N,$$
$$D^1_{00} = -D^1_{NN} = N(N+1)/4.$$

Problem 2 Use `CODE DM.1` and `CODE DM.2` to compute D^1 with the Legendre-Gauss-Lobatto points, with $N = 5$. Compare your results with the direct formulas given in Problem 1.

2.2 Differentiation matrices for Fourier collocation methods

Fourier series and differentiation
Differentiation matrices using direct formulas

The discussions in Section 2.1 are concentrated on the algebraic polynomial basis functions. Another class of basis functions is the trigonometric functions which are more suitable for representing *periodic* phenomena. For convenience, let us assume that the function being interpolated is periodic with period 2π.

It is known from Section 1.5 that the functions E_k defined by $E_k(x) = e^{ikx}$ form an orthogonal system of functions in the complex space $L_2(0, 2\pi)$. An *exponential polynomial* of degree at most n is any function of the form

$$\begin{aligned} p(x) &= \sum_{k=0}^{n} d_k e^{ikx} = \sum_{k=0}^{n} d_k E_k(x) \\ &= \sum_{k=0}^{n} d_k (e^{ix})^k . \end{aligned} \tag{2.2.1}$$

The last expression in this equation explains the source of the terminology because it shows p to be a polynomial of degree $\leqslant n$ in the variable e^{ix}.

Lemma 2.2.1 *The exponential polynomial that interpolates a prescribed function f at $x_j = 2\pi j/N$, $0 \leqslant j \leqslant N-1$, is given by*

$$P(x) = \sum_{k=0}^{N-1} c_k E_k(x), \qquad \textit{with} \quad c_k = \langle f, E_k \rangle_N, \tag{2.2.2}$$

where the inner product is defined by (1.5.2). *In other words,* $P(x_j) = f(x_j)$, $0 \leqslant j \leqslant N-1$.

Proof The above result can be obtained by the direct calculations: for $0 \leqslant m \leqslant N-1$,

$$\begin{aligned} P(x_m) &= \sum_{k=0}^{N-1} c_k E_k(x_m) = \sum_{k=0}^{N-1} \frac{1}{N} \sum_{j=0}^{N-1} f(x_j)\overline{E_k(x_j)} E_k(x_m) \\ &= \sum_{j=0}^{N-1} f(x_j) \langle E_m, E_j \rangle_N = f(x_m)\,, \end{aligned}$$

where in the last step we have used (1.5.4) and the fact $0 \leqslant |m-j| < N$. □

Fourier series and differentiation

It is well known that if f is 2π-periodic and has a continuous first derivative then its Fourier series converges uniformly to f. In applications, we truncate the infinite Fourier series

$$\mathcal{F}(x) \sim \sum_{k=-\infty}^{\infty} \hat{f}(k) e^{ikx}, \tag{2.2.3}$$

to the following finite series:

$$F(x) = \sum_{k=-N/2}^{N/2-1} \alpha_k e^{ikx}. \tag{2.2.4}$$

Assume that $F(x_j) = f(x_j)$, where $x_j = 2\pi j/N, 0 \leqslant j \leqslant N-1$. It can be shown that

$$f(x_j) = \sum_{k'=0}^{N-1} \alpha_{k'-N/2} (-1)^j e^{ik' x_j}, \qquad 0 \leqslant j \leqslant N-1.$$

This, together with (2.2.2), gives

$$\alpha_{k'-N/2} = \frac{1}{N} \sum_{j=0}^{N-1} (-1)^j F(x_j) e^{-ik' x_j}, \qquad 0 \leqslant k' \leqslant N-1. \tag{2.2.5}$$

We now differentiate the truncated Fourier series $F(x)$ termwise to get the approximate derivatives. It follows from (2.2.4) that

$$F^{(m)}(x) = \sum_{k=-N/2}^{N/2} \alpha_k (ik)^m e^{ikx},$$

where m is a positive integer. We can write $F^{(m)}(x)$ in the following equivalent form:

$$F^{(m)}(x) = \sum_{k'=0}^{N-1} \alpha_{k'-N/2} \Big(i(k' - N/2) \Big)^m e^{i(k'-N/2)x}. \tag{2.2.6}$$

Using (2.2.5) and (2.2.6), we obtain

$$\begin{pmatrix} F^{(m)}(x_0) \\ F^{(m)}(x_1) \\ \vdots \\ F^{(m)}(x_{N-1}) \end{pmatrix} = \Big((-1)^j e^{ikx_j} (i(k - N/2))^m \Big)_{j,k=0}^{N-1} \begin{pmatrix} \alpha_{-N/2} \\ \alpha_{-N/2+1} \\ \vdots \\ \alpha_{N/2-1} \end{pmatrix}$$

$$= \frac{1}{N} \Big((-1)^j e^{ikx_j} (i(k - N/2))^m \Big)_{j,k=0}^{N-1} \Big((-1)^k e^{-ijx_k} \Big)_{j,k=0}^{N-1} \begin{pmatrix} F(x_0) \\ F(x_1) \\ \vdots \\ F(x_{N-1}) \end{pmatrix}.$$

This indicates that the m-th order differentiation matrix associated with Fourier spectral-collocation methods is given by

$$D^m = \frac{1}{N} \Big((-1)^j e^{ikx_j} (i(k - N/2))^m \Big)_{j,k=0}^{N-1} \Big((-1)^k e^{-ijx_k} \Big)_{j,k=0}^{N-1} \tag{2.2.7}$$

A pseudocode for computing D^m is given below:

```
CODE FPS.1
function Dm=FDMx(m,N)
%collocation points:  x(j)=2*π*j/N, 1⩽j⩽N-1
for j=0 to N-1
    for k=0 to N-1
        A(j,k)=(-1)^j*exp(i*k*x(j))*(i*(k-N/2))^m
        B(j,k)=(-1)^k*exp(-i*j*x(k))
    endfor
endfor
Dm=(1/N)*A*B
```

To test the above code, we consider a simple example. Let $f(x) = 1/(2 + \sin x)$. Let $\mathbf{F} = (f(x_0), f(x_1), \cdots, f(x_{N-1}))^{\mathrm{T}}$ with $x_j = 2\pi j/N$. The matrices $D1 =$

`FDMx(1,N)` and $D2 =$ `FDMx(2,N)` are given by `CODE FPS.1`. We plot the L^1 errors

$$\texttt{err1} = \frac{1}{N}\sum_{j=0}^{N-1} |(D1 * \mathbf{F})_j - f'(x_j)|, \quad \texttt{err2} = \frac{1}{N}\sum_{j=0}^{N-1} |(D2 * \mathbf{F})_j - f''(x_j)|$$

in Fig. 2.1 (a). It is observed that the above L^1 errors decay to zero very rapidly.

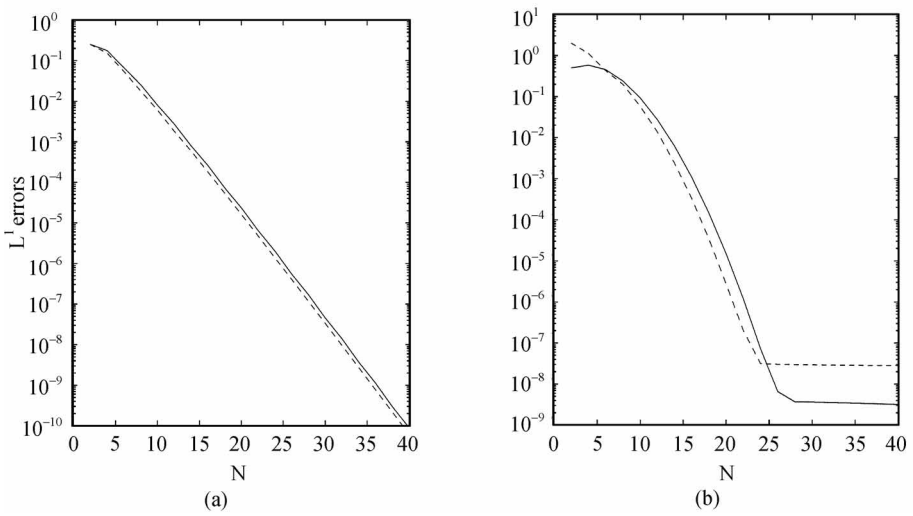

Figure 2.1

(a) $f(x) = 1/(2+\sin x)$; (b) $f(x) = e^{-2(x-\pi)^2}$. The solid line is for `err1`, and the dashed line is for `err2`

It should be pointed out that the convergence holds only for periodic functions. If we change the above $f(x)$ to a non-periodic function, say $f(x) = x^2$, then the errors `err1` and `err2` defined above will diverge to infinity as N becomes large.

Apart from the periodic functions, the Fourier spectral methods can also handle functions which decay to zero away from a finite interval. We can always use a linear transform to change the finite interval to $[0, 2\pi]$. To see this, we consider $f(x) = e^{-2(x-\pi)^2}$. In Fig. 2.1(b), we plot `err1` and `err2` for this function. It is noted that the errors will not decrease after a critical value of N, but the errors for large N will be of the same magnitudes of $f'(x)$ and $f''(x)$ away from $[0, 2\pi]$.

Differentiation matrices using direct formulas

Again choose $x_j = 2\pi j/N, 0 \leqslant j \leqslant N-1$. The corresponding interpolant is

given by (Gottlieb et al.[36]; Henrici [81], Section 13.6)

$$t_N(x) = \sum_{j=1}^{N} \phi_j(x) f_j,$$

where the Lagrange polynomials $F_j(x)$ are of the form

$$F_j(x) = \frac{1}{N} \sin \frac{N}{2}(x - x_j) \cot \frac{1}{2}(x - x_j), \qquad N \text{ even}, \tag{2.2.8a}$$

$$F_j(x) = \frac{1}{N} \sin \frac{N}{2}(x - x_j) \csc \frac{1}{2}(x - x_j), \qquad N \text{ odd}. \tag{2.2.8b}$$

It can be shown that an equivalent form of $t_N(x)$ (barycentric form of interpolant) is (see Henrici[81], Section 13.6):

$$t_N(x) = \sum_{j=1}^{N} (-1)^j f_j \cot \frac{1}{2}(x - x_j) \Big/ \sum_{j=1}^{N} (-1)^j \cot \frac{1}{2}(x - x_j), \qquad N \text{ even}, \tag{2.2.9a}$$

$$t_N(x) = \sum_{j=1}^{N} (-1)^j f_j \csc \frac{1}{2}(x - x_j) \Big/ \sum_{j=1}^{N} (-1)^j \csc \frac{1}{2}(x - x_j), \qquad N \text{ odd}. \tag{2.2.9b}$$

The differentiation matrix $D^{(m)} = (F_j^{(m)}(x_k))$ is obtained by Gottlieb et al.. For N even, $1 \leqslant k, j \leqslant N$:

$$D^1_{kj} = \begin{cases} 0 & \text{if } k = j, \\ \dfrac{1}{2}(-1)^{k-j} \cot \dfrac{(k-j)h}{2} & \text{if } k \neq j, \end{cases} \tag{2.2.10a}$$

$$D^2_{kj} = \begin{cases} -\dfrac{\pi^2}{3h^2} - \dfrac{1}{6} & \text{if } k = j, \\ -(-1)^{k-j} \dfrac{1}{2} \csc^2 \dfrac{(k-j)h}{2} & \text{if } k \neq j. \end{cases} \tag{2.2.10b}$$

Similarly, for N odd, $1 \leqslant k, j \leqslant N$:

$$D^1_{kj} = \begin{cases} 0 & \text{if } k = j, \\ \dfrac{1}{2}(-1)^{k-j} \csc \dfrac{(k-j)h}{2} & \text{if } k \neq j, \end{cases} \tag{2.2.11a}$$

$$D^2_{kj} = \begin{cases} -\dfrac{\pi^2}{3h^2} - \dfrac{1}{12} & \text{if } k = j, \\ -(-1)^{k-j}\dfrac{1}{2}\csc\dfrac{(k-j)h}{2}\cot\dfrac{(k-j)h}{2} & \text{if } k \neq j. \end{cases} \tag{2.2.11b}$$

It can be shown that if N is odd then

$$D^m = (D^1)^m. \tag{2.2.12}$$

If N is even, the above formula only holds for odd m.

Exercise 2.2

Problem 1 Use `CODE FPS.1` to compute D^2 and D^3.

a. Let $N = 6$ and $m = 3$. Verify (2.2.12) by using (2.2.10a). Show also that (2.2.12) does not hold for $m = 2$.

b. Let $N = 5$ and $m = 2, 3$. Verify (2.2.12) by using (2.2.11a).

Problem 2 Design an algorithm to compute the differentiation matrix D^1 for the Chebyshev collocation method that uses FFT.

Problem 3 Consider the eigenvalue problem

$$-u'' + x^2 u = \lambda u, \qquad x \in \mathbb{R}.$$

This problem is related to a quantum harmonic oscillator, whose eigenvalues are $\lambda = 1, 3, 5, \cdots$ and the eigenfunctions u are the Hermite functions $e^{-x^2/2}H_n(x)$. Since these solutions decay rapidly, for practical computations we can truncate the infinite spatial domain to the periodic domain $[-L, L]$, provided L is sufficiently large. Using a Fourier collocation method to find the first 4 eigenvalues, with $N = 6, 12, 18, 24$ and 36. ①

2.3 Eigenvalues of Chebyshev collocation operators

Advection operator
Diffusion operator with Dirichlet boundary conditions
Diffusion operator with Neumann boundary conditions
Comparison with finite difference methods

The eigenvalues of a matrix A are the complex numbers λ for which the matrix $A - \lambda I$

① Hint: MATLAB has a code for finding the eigenvalues of $Av = \lambda v$.

is not invertible. The *spectral radius* of A is defined by the equation

$$\rho(A) = \max\{|\lambda| : det(A - \lambda I) = 0\}.$$

Thus, $\rho(A)$ is the smallest number such that a circle with that radius centered at 0 in the complex plane will contain all the eigenvalues of A.

Using spectral methods to deal with time-dependent differential equations will often result in a system of ODEs. For example, consider the linear heat equation $u_t = u_{xx}$ with appropriate initial and boundary conditions. If we use collocation methods, we will obtain a system of ODE like $U'(t) = AU + b$, where the matrix A is related to the second order differentiation matrix investigated in Section 2.1, the vector b is related to the boundary conditions. Now the spectral radius of A is important: it determines the maximum time step allowed by using an explicit scheme for this ODE system through the relation $\Delta t \rho(A) \lesssim 1$.

The condition number of a matrix A is defined by

$$\kappa(A) = \max\{|\lambda| : \det(A - \lambda I) = 0\} / \min\{|\lambda| : \det(A - \lambda I) = 0\}.$$

A matrix with a large condition number is said to be *ill conditioned* while the matrix is said to be well conditioned if the condition number of A is of moderate size. There are two main numerical difficulties in dealing with Ill-conditioned matrices, first of all, the solution of $Ax = b$ is very sensitive to small changes in the vector b if A is ill conditioned; secondly, the number of iterations needed for solving $Ax = b$ using an iterative method usually increases with the condition number of A.

Using spectral methods to solve differential equations will often require solving a system of algebraic equations. In this case, information about the underlying matrix such as spectral radius and condition number will be very useful. As we shall see in Section 2.4, the underlying matrix is often formed by the differentiation matrices. Therefore, it is helpful to study the eigenvalues of the differentiation matrices associated with different spectral methods. In this section, we will investigate the eigenvalues of the Chebyshev collocation operators. Some references related to this section can be found in [164], [169].

Advection operator

We consider here the advection operator

$$Lu = \frac{du}{dx}, \qquad x \in (-1, 1), \tag{2.3.1}$$

subject to the boundary condition $u(1) = 0$. We use the Chebyshev collocation method with the collocation points $x_j = \cos(\pi j/N)$. The eigenvalues of the collocation operator are defined by the set of equations

$$\frac{dU(x_j)}{dx} = \lambda U(x_j), \quad 1 \leqslant j \leqslant N; \quad U(x_0) = 0, \tag{2.3.2}$$

provided U is a non-trivial polynomial of degree N. It can be shown theoretically that the real parts of λ are strictly negative, while the modulus satisfies a bound of the form $|\lambda| \lesssim N^2$. We will verify this by numerical experiments.

Since $U(x_0) = 0$, it is easy to see that (2.3.2) leads to a standard eigenvalue problem $AU = \lambda U$, where A is formed by removing the first column and the first row from $D1$, where $D1$ is given by `CODE DM.3` in Section 2.1.

```
CODE Eigen.1
Input N
%first order differentiation matrix
call CODE DM.3 in Sect 2.1 to get D1(i,j), 0⩽i,j⩽N
%form the coefficient matrix:  A(i,j)=D1(i,j), 1⩽i,j⩽N
compute the eigenvalues of A
find the largest and smallest |λ|
ρ(A)=the largest |λ|; κ(A)=ρ(A)/the smallest |λ|
```

In MATLAB, `eig(A)` is the vector containing the eigenvalues of matrix A; `max(abs(eig(A)))` gives the spectral radius of A; `min(abs(eig(A)))` gives the smallest $|\lambda|$ of A. Numerical results show that the real parts of `eig(A)` are strictly negative, and that

$$\rho(A) \leqslant 0.5N^2, \qquad \kappa(A) \leqslant N^2, \tag{2.3.3}$$

as can be seen from Fig. 2.2.

Diffusion operator with Dirichlet boundary conditions

We now consider the diffusion operator

$$Lu = \frac{d^2u}{dx^2}, \qquad x \in (-1, 1), \tag{2.3.4}$$

with homogeneous Dirichlet boundary conditions, i.e., $u(\pm 1) = 0$. The eigenvalues of the Chebyshev-collocation approximation to this operator are defined by the set of

equations

$$\begin{aligned} &\frac{d^2U(x_j)}{dx^2} = \lambda U(x_j), \qquad 1 \leqslant j \leqslant N-1, \\ &U(x_0) = 0, \quad U(x_N) = 0, \end{aligned} \tag{2.3.5}$$

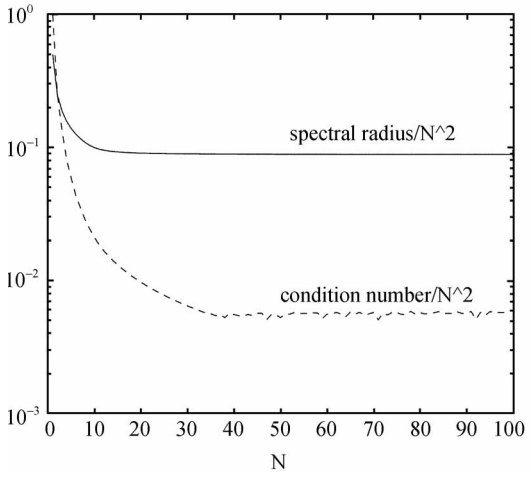

Figure 2.2 The spectral radius and condition number associated with the advection operator.

where $\{x_j\}$ are the Chebyshev-Gauss-Lobatto points and U is a polynomial of degree N. It was shown in [58] that there exist two positive constants c_1, c_2 independent of N such that

$$0 < c_1 \leqslant -\lambda \leqslant c_2 N^4. \tag{2.3.6}$$

We will verify this by numerical experiments with the following code:

```
CODE Eigen.2
%Zero Dirichlet boundary conditions
Input N
%first order differentiation matrix
call CODE DM.3 in Section 2.1 to get D1(i,j), 0⩽i,j⩽N
D2=D1*D1
%form the coefficient matrix:  A(i,j)=D2(i,j), 1⩽i,j⩽N-1
compute the eigenvalues of A
find the largest and smallest |λ|
ρ(A)=the largest |λ|; κ(A)=ρ(A)/the smallest |λ|
```

In Fig. 2.3, we plot the spectral radius and condition number for the Dirichlet prob-

lem. It can be seen from Fig. 2.3 that

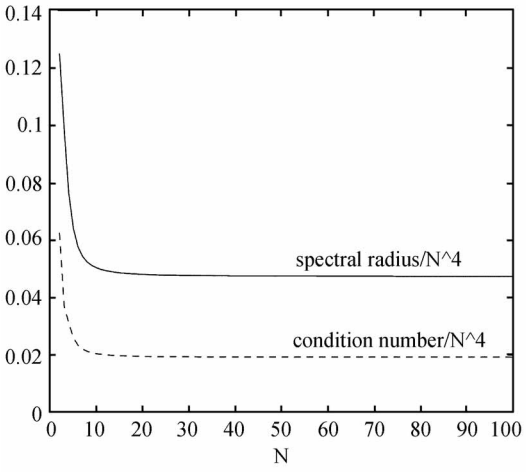

Figure 2.3 The spectral radius and condition number associated with the Chebyshev spectral methods.

$$\begin{aligned} \rho(A) &\approx 0.047N^4, \qquad \text{for} \quad N \geqslant 30, \\ \kappa(A) &\approx 0.019N^4, \qquad \text{for} \quad N \geqslant 15. \end{aligned} \tag{2.3.7}$$

It is also observed from the numerical results that

$$\min|\lambda| \approx 2.467, \qquad \text{for} \quad N \geqslant 5. \tag{2.3.8}$$

Diffusion operator with Neumann boundary conditions

We now consider the diffusion operator (2.3.4) with the homogeneous Neumann boundary conditions $u'(\pm 1) = 0$. The eigenvalues of the Chebyshev-collocation approximation to this operator are defined by the set of equations

$$\begin{aligned} &\frac{d^2U(x_j)}{dx^2} = \lambda U(x_j), \qquad 1 \leqslant j \leqslant N-1, \\ &U'(x_0) = 0, \quad U'(x_N) = 0, \end{aligned} \tag{2.3.9}$$

where, once again, $\{x_j\}$ are the Chebyshev-Gauss-Lobatto points and U is a polynomial of degree N. We follow the procedure in Section 2.4 to form the corresponding matrix. Our boundary conditions are of the type (2.4.2) with $a_+ = b_- = 0, b_+ = a_- = 1, c_- = c_+ = 0$. Using (2.4.13), the coefficient matrix $A = (a_{ij})$ is given by

$$a_{ij} = (D^2)_{ij} - (D^2)_{i0}\tilde{\alpha}_{0j} - (D^2)_{iN}\tilde{\alpha}_{Nj}, \qquad 1 \leqslant i, j \leqslant N-1,$$

where

$$\tilde{\alpha}_{0j} = \Big((D^1)_{0N}(D^1)_{Nj} - (D^1)_{NN}(D^1)_{0j}\Big) \cdot \Big((D^1)_{N0}(D^1)_{0N} - (D^1)_{00}(D^1)_{NN}\Big)^{-1},$$

$$\tilde{\alpha}_{Nj} = \Big((D^1)_{N0}(D^1)_{0j} - (D^1)_{00}(D^1)_{Nj}\Big) \cdot \Big((D^1)_{N0}(D^1)_{0N} - (D^1)_{00}(D^1)_{NN}\Big)^{-1}.$$

A pseudocode for computing the spectral radius and condition number of the Neumann problem is given below.

```
CODE Eigen.3
%Zeor Neumann boundary conditions
Input N
%first order differentiation matrix
call CODE DM.3 in Sect 2.1 to get D1(i,j), 0⩽i,j⩽N
D2=D1*D1
%form the coefficient matrix
ss=D1(N,0)*D1(0,N)-D1(0,0)*D1(N,N)
for j=1 to N-1 do
  α̃0j=(D1(0,N)*D1(N,j)-D1(N,N)*D1(0,j))/ss
  α̃Nj=(D1(N,0)*D1(0,j)-D1(0,0)*D1(N,j))/ss
    for i=1 to N-1 do
      A(i,j)=D2(i,j)-D2(i,0)*α̃0j-D2(i,N)*α̃Nj
    endfor
endfor
Compute the eigenvalues of A
Calculate the spectral radius of A and the condition number
```

Numerical results show that, except for the zero eigenvalue, the following inequalities hold:

$$2.18 \leqslant -\lambda \leqslant 0.03N^4. \tag{2.3.10}$$

Also, it is observed that

$$\rho(A) \approx 0.014N^4, \qquad \text{for} \quad N \geqslant 20. \tag{2.3.11}$$

Comparison with finite difference methods

Let us first consider the eigenvalues of the following tridiagonal matrix:

$$D = \begin{pmatrix} a & c & & & \\ b & a & c & & \\ & \ddots & \ddots & \ddots & \\ & & b & a & c \\ & & & b & a \end{pmatrix} \qquad \text{with} \quad b \cdot c \neq 0. \tag{2.3.12}$$

The size of the matrix D is $(N-1) \times (N-1)$. We will show that the eigenvalues of D are given by

$$\lambda_k = a + 2\sqrt{bc}\cos(\pi k/N), \qquad 1 \leqslant k \leqslant N-1. \tag{2.3.13}$$

By definition, the eigenvalues of D satisfy

$$DV = \lambda V, \tag{2.3.14}$$

where V is the eigenvector associated with λ. Equivalently, (2.3.14) can be written as

$$bV_{j-1} + aV_j + cV_{j+1} = \lambda V_j, \qquad 1 \leqslant j \leqslant N-1, \tag{2.3.15a}$$

$$V_0 = 0, \quad V_N = 0. \tag{2.3.15b}$$

In analogy to solving a second-order ODE with constant coefficients, we assume a special form of V_j, namely $V_j = \beta^j$, where $\beta \neq 0$ is a constant to be determined. Substituting this into (2.3.15a) gives

$$b + a\beta + c\beta^2 = \lambda\beta. \tag{2.3.16}$$

Since $b \cdot c \neq 0$, the above quadratic equation has two roots β_1 and β_2. If $\beta_1 \neq \beta_2$, then the general solution of (2.3.15a) is given by

$$V_j = c_1\beta_1^j + c_2\beta_2^j, \qquad 1 \leqslant j \leqslant N-1, \tag{2.3.17}$$

where c_1 and c_2 are two constants. It follows from (2.3.15b) that $c_1 + c_2 = 0$ and $c_1\beta_1^N + c_2\beta_2^N = 0$, which yields $(\beta_1/\beta_2)^N = 1$. Therefore, we obtain

$$\frac{\beta_1}{\beta_2} = e^{i2\pi k/N}, \qquad 1 \leqslant k \leqslant N-1. \tag{2.3.18}$$

Since β_1 and β_2 are roots of (2.3.16), we have

$$\beta_1 + \beta_2 = -(a-\lambda)/c, \quad \beta_1\beta_2 = b/c. \tag{2.3.19}$$

Combining (2.3.18) and the second equation of (2.3.19) gives

$$\beta_1 = \sqrt{\frac{b}{c}} e^{i\pi k/N}, \qquad \beta_2 = \sqrt{\frac{b}{c}} e^{-i\pi k/N}.$$

This, together with the first equation of (2.3.19), leads to (2.3.13).

We now consider the central-difference method for the diffusion operator (2.3.4) with homogeneous (Dirichlet) boundary conditions. In this case, the corresponding eigenvalue problem is $AV = \lambda V$, where A is a tridiagonal matrix of the form (2.3.12), with $a = -2N^2, b = c = N^2$. By (2.3.13), we see that the eigenvalues of A satisfy

$$\max|\lambda| \approx 4N^2, \qquad \min|\lambda| \approx \pi^2\,. \tag{2.3.20}$$

The above results indicate that the spectral radius and condition number of the Chebyshev collocation method for first- and second-order operators grow like N^2 and N^4 respectively, while those of the finite difference method (at equally spaced points) grow like N and N^2 respectively. This rapid growth in spectral radius and condition number of the Chebyshev collocation method is due to the fact that the smallest distance between neighboring collocation points behave like N^{-2} near the boundaries. While this clustering of the collocation points near the boundaries provide extra resolution for problems with thin boundary layers which are present in many physical situations, it does lead to severe time step restrictions if an explicit scheme is used. Therefore, it is advised that second or higher derivative operators should be treated implicitly to allow reasonable time steps.

Exercise 2.3

Problem 1 By computing $\lambda_{\max}$ for $N = 30, 40, 50, 60$ and 70, show that in Chebyshev collocation method (using Gauss-Lobatto points) the growth of second-derivative eigenvalues behaves like

$$\begin{aligned} \lambda_{\max} &\approx -0.047N^4 \quad \text{Dirichlet}, \qquad N \geqslant 30, \\ \lambda_{\max} &\approx -0.014N^4 \quad \text{Neumann}, \qquad N \geqslant 30. \end{aligned}$$

Problem 2 What will be the corresponding growth of third-derivative eigenvalues? Verify your results numerically.

2.4 Chebyshev collocation method for two-point BVPs

BVPs with Dirichlet boundary conditions
BVPs with general boundary conditions
Numerical experiments

In this section, we introduce the Chebyshev collocation method for the linear second-order two-point boundary-value problem (BVP),

$$\epsilon u''(x) + p(x)u'(x) + q(x)u(x) = f(x), \quad x \in I := (-1, 1), \tag{2.4.1}$$

where ϵ is a (fixed) parameter that controls the singular behavior of the problem, and p, q and f are given functions. If ϵ is of order 1, the problem is non-singular, while for sufficiently small ϵ, the problem may exhibit singular behavior such as sharp boundary and interior layers. In the latter case, the problem (2.4.1) is called a singularly perturbed BVP. The boundary condition for (2.4.1) is given by

$$a_-u(-1) + b_-u'(-1) = c_-, \qquad a_+u(1) + b_+u'(1) = c_+, \tag{2.4.2}$$

Without loss of generality, we assume $a_\pm \geqslant 0$. We also assume

$$\begin{aligned} &a_-^2 + b_-^2 \neq 0, \ \text{ and } a_-b_- \leqslant 0; \ a_+^2 + b_+^2 \neq 0, \ \text{ and } a_+b_+ \geqslant 0; \\ &q(x) - \frac{1}{2}p'(x) \geqslant 0, \ \ \forall x \in (I); \\ &p(1) > 0 \text{ if } b_+ \neq 0, \ p(-1) < 0 \text{ if } b_- \neq 0. \end{aligned} \tag{2.4.3}$$

It is easy to check that the above conditions ensure the well-posedness of (2.4.1)and (2.4.2).

We now discuss how to solve the problem with $\epsilon = \mathcal{O}(1)$ by using the Chebyshev collocation method. The case with $0 < \epsilon \ll 1$ will be considered in Section 5.1.

BVPs with Dirichlet boundary conditions

We first consider the simplest boundary conditions:

$$u(-1) = c_-, \qquad u(1) = c_+. \tag{2.4.4}$$

The Chebyshev interpolation polynomial can be written as

$$u^N(x) = \sum_{j=0}^{N} U_j F_j(x), \tag{2.4.5}$$

where $x_j = \cos(j\pi/N)$, $0 \leqslant j \leqslant N$ are the Chebyshev-Gauss-Lobatto collocation points, $\{U_j\}_{j=1}^{N-1}$ are the unknown coefficients to be determined, and $F_j(x)$ is the Lagrange interpolation polynomial associated with $\{x_j\}$. The Chebyshev collocation method is to seek u^N in the form of (2.4.5) such that $u^N(-1) = c_-$, $u^N(1) = c_+$,

and that the equation holds at the interior collocation points:

$$\epsilon u_{xx}^N(x_j) + p(x_j)u_x^N(x_j) + q(x_j)u^N(x_j) = f(x_j), \qquad 1 \leqslant j \leqslant N-1. \quad (2.4.6)$$

Now using the definition of the differentiation matrix introduced in the Section 2.1, we obtain a system of linear equations,

$$\begin{aligned} &\sum_{j=1}^{N-1} \left[\epsilon(D^2)_{ij} + p(x_i)(D^1)_{ij} + q(x_i)\delta_{ij}\right] U_j \\ &= f(x_i) - \left[\epsilon(D^2)_{i0} + p(x_i)(D^1)_{i0}\right] c_+ - \left[\epsilon(D^2)_{iN} + p(x_i)(D^1)_{iN}\right] c_-, \end{aligned} \quad (2.4.7)$$

for the $\{U_j\}_{j=1}^{N-1}$, where δ_{ij} is the Kronecker delta. In the above equations, we have used the boundary conditions $U_0 = c_+, U_N = c_-$ (notice that $x_0 = 1$ and $x_N = -1$).

To summarize: the spectral-collocation solution for the BVP (2.4.1) with the Dirichlet boundary conditions (2.4.4) satisfies the linear system

$$A\bar{U} = \bar{b}, \quad (2.4.8)$$

where $\bar{U} = [U_1, \cdots, U_{N-1}]^{\mathrm{T}}$; the matrix $A = (a_{ij})$ and the vector $\bar{b}$ are given by

$$\begin{aligned} a_{ij} &= \epsilon(D^2)_{ij} + p(x_i)(D^1)_{ij} + q(x_i)\delta_{ij}, \qquad 1 \leqslant i, j \leqslant N-1, \\ b_i &= f(x_i) - \left[\epsilon(D^2)_{i0} + p(x_i)(D^1)_{i0}\right] c_+ - [\epsilon(D^2)_{iN} \\ &\quad + p(x_i)(D^1)_{iN}]c_-, \qquad 1 \leqslant i \leqslant N-1. \end{aligned} \quad (2.4.9)$$

The solution to the above system gives the approximate solution to (2.4.1) and (2.4.4) at the collocation points. The approximation solution in the whole interval is determined by (2.4.5). A pseudo-code is given below:

```
CODE PSBVP.1
Input N, ϵ, p(x), q(x), f(x), c_-, c_+
%collocation points:  x(j)=cos(πj/N), 0⩽j⩽N
%first order differentiation matrix
call CODE DM.3 in Sect 2.1 to get D1
%compute second order differentiation matrix:  D2=D1*D1
% compute the stiffness matrix A
for i=1 to N-1 do
      for j=1 to N-1 do
          if i=j  A(i,j)=ϵ*D2(i,j)+p(x(i))*D1(i,j)+q(x(i))
          else A(i,j)=ϵ*D2(i,j)+p(x(i))*D1(i,j)
          endif
```

```
      endfor
% compute the right side vector b
   ss1=ϵ*D2(i,0)+p(x(i))*D1(i,0); ss2=ϵ*D2(i,N)+p(x(i))
   *D1(i,N)
   b(i)=f(i)-ss1*c+-ss2*c−
endfor
% solve the linear system to get the unknown vector
u=A^{-1}b
Output u(1), u(2), ···, u(N-1)
```

A MATLAB code is also provided below:

```
CODE PSBVP.2
Input N, eps, p(x), q(x), f(x), cminus, cplus
j=[1:1:N-1];  x=[cos(pi*j/N)]';
D1=DM1(N);  D2=D1^2;
for i=1:N-1
   s=x(i); p1=p(s); q1=q(s); f1=f(s);
      for j=1:N-1
        if i==j
          A(i,j)=eps*D2(i+1,j+1)+p1*D1(i+1,j+1)+q1;
        else
          A(i,j)=eps*D2(i+1,j+1)+p1*D1(i+1,j+1);
        end
      end
   ss1=eps*D2(i+1,1)+p1*D1(i+1,1);
   ss2=eps*D2(i+1,N+1)+p1*D1(i+1,N+1);
   b(i)=f1-ss1*cplus-ss2*cminus;
end
u=A\b';
```

For test problems having exact solutions, a few more lines may be added to compute the maximum errors:

```
%if the exact solution uexact(x) is given
for i=1:N-1
   error(i)=abs(u(i)-uexact(i));
end
xx=N; err=max(error);
fprintf(1,      '%16.0f          %13.3e  \n', [xx;  err]);
```

The above MATLAB code will be used to compute the numerical solutions for Example 2.4.1 in this section and Example 5.1.1 in Section 5.1.

BVPs with general boundary conditions

We now consider the general boundary conditions (2.4.2). Without loss of generality, we assume $b_- \neq 0$ and $b_+ \neq 0$ (otherwise we will have simpler cases). It follows from (2.4.2) that

$$a_- U_N + b_- \sum_{j=0}^{N} (D^1)_{Nj} U_j = c_-, \qquad a_+ U_0 + b_+ \sum_{j=0}^{N} (D^1)_{0j} U_j = c_+,$$

which leads to

$$\begin{aligned} &b_-(D^1)_{N0}U_0 + \Big(a_- + b_-(D^1)_{NN}\Big)U_N = c_- - b_- \sum_{j=1}^{N-1} (D^1)_{Nj} U_j, \\ &\Big(a_+ + b_+(D^1)_{00}\Big)U_0 + b_+(D^1)_{0N}U_N = c_+ - b_+ \sum_{j=1}^{N-1} (D^1)_{0j} U_j. \end{aligned} \tag{2.4.10}$$

Solving the above equations we find

$$U_0 = \tilde{c}_+ - \sum_{j=1}^{N-1} \tilde{\alpha}_{0j} U_j, \qquad U_N = \tilde{c}_- - \sum_{j=1}^{N-1} \tilde{\alpha}_{Nj} U_j, \tag{2.4.11}$$

where the parameters $\tilde{c}_+, \tilde{\alpha}_{0j}, \tilde{c}_-, \tilde{\alpha}_{Nj}$ are defined by

$$\begin{aligned} &\tilde{c}_+ = \Big(\tilde{d}c_- - \tilde{b}c_+\Big)/(\tilde{a}\tilde{d} - \tilde{c}\tilde{b}), \qquad \tilde{c}_- = \Big(\tilde{a}c_+ - \tilde{c}c_-\Big)/(\tilde{a}\tilde{d} - \tilde{c}\tilde{b}), \\ &\tilde{\alpha}_{0j} = \Big(\tilde{d}b_-(D^1)_{Nj} - \tilde{b}b_+(D^1)_{0j}\Big)/(\tilde{a}\tilde{d} - \tilde{c}\tilde{b}), \\ &\tilde{\alpha}_{Nj} = \Big(\tilde{a}b_+(D^1)_{0j} - \tilde{c}b_-(D^1)_{Nj}\Big)/(\tilde{a}\tilde{d} - \tilde{c}\tilde{b}), \\ &\tilde{a} := b_-(D^1)_{N0}, \qquad \tilde{b} := a_- + b_-(D^1)_{NN}, \\ &\tilde{c} := a_+ + b_+(D^1)_{00}, \qquad \tilde{d} := b_+(D^1)_{0N}. \end{aligned}$$

To summarize: let the constants b_- and b_+ in (2.4.2) be nonzero. The spectral-collocation solution for the BVP (2.4.1) with the general boundary condition (2.4.2) satisfies the linear system

$$A\bar{U} = \bar{b}, \tag{2.4.12}$$

where $A = (a_{ij})$ is a $(N-1)\times(N-1)$ matrix and $\bar{b} = (b_j)$ is a $(N-1)$-dimensional

vector:

$$a_{ij} = \epsilon(D^2)_{ij} + p(x_i)(D^1)_{ij} + q_i\delta_{ij} - \left[\epsilon(D^2)_{i0} + p(x_i)(D^1)_{i0}\right]\tilde{\alpha}_{0j} - \left[\epsilon(D^2)_{iN} + p(x_i)(D^1)_{iN}\right]\tilde{\alpha}_{Nj},$$
$$b_i = f(x_i) - \left[\epsilon(D^2)_{i0} + p(x_i)(D^1)_{i0}\right]\tilde{c}_+ - \left[\epsilon(D^2)_{iN} + p(x_i)(D^1)_{iN}\right]\tilde{c}_-. \tag{2.4.13}$$

A pseudo-code is given below:

```
CODE PSBVP.3
Input N, ϵ, p(x), q(x), f(x), c_-, c_+, a_-, b_-, a_+, b_+
%collocation points:  x(j)=cos(πj/N), 0⩽j⩽N
%first order differentiation matrix
call CODE DM.3 in Sect 2.1 to get D1
%compute second order ifferentiation matrix
D2=D1*D1
% calculate some constants
ta=a_+*D1(N,0); tb=a_-+a_+*D1(N,N)
tc=b_-+b_+*D1(0,0); td=b_+*D1(0,N); te=ta*td-tc*tb
c̃_+=(td*c_--tb*c_+)/te; c̃_-=(ta*c_+-tc*c_-)/te
% compute the stiffness matrix A
for i=1 to N-1 do
  ss1=ϵ*D2(i,0)+p(x(i))*D1(i,0); ss2=ϵ*D2(i,N)+p(x(i))
  *D1(i,N)
      for j=1 to N-1 do
         ss3=(td*a_+*D1(N,j)-tb*b_+*D1(0,j))/te
         ss4=(ta*b_+*D1(0,j)-tc*a_+*D1(N,j))/te
         ss5=ss1*ss3+ss2*ss4
            if i=j  A(i,j)=ϵ*D2(i,j)+p(x(i))*D1(i,j)
            +q(x(i))-ss5
            else A(i,j)=ϵ*D2(i,j)+p(x(i))*D1(i,j)-ss5
            endif
      endfor
%compute the right side vector b:  b(i)=f(i)-ss1*c̃_+-ss2*c̃_-
endfor
% solve the linear system to get the unknown vector
u=A^{-1}b
Output u(1), u(2), ···, u(N-1)
```

Numerical experiments

In this subsection we will consider two numerical examples. The numerical re-

sults will be obtained by using `CODE PSBVP.2` and `CODE PSBVP.3`, respectively.

Example 2.4.1 *Consider the following problem*

$$u''+xu'(x)-u(x)=(24+5x)e^{5x}+(2+2x^2)\cos(x^2)-(4x^2+1)\sin(x^2), \quad (2.4.14)$$
$$u(-1)=e^{-5}+\sin(1), \quad u(1)=e^{5}+\sin(1).$$

The exact solution for Example 2.4.1 is $u(x) = e^{5x} + \sin(x^2)$. We solve this problem by using different values of N and compute the maximum error which is defined by $\max_{1\leqslant j\leqslant N-1}|U_j - u(x_j)|$. It is the maximum error at the interior collocation points. Here is the output.

N	Maximum error	N	Maximum error
5	2.828e+00	13	6.236e-06
6	8.628e-01	14	9.160e-07
7	1.974e-01	15	1.280e-07
8	3.464e-02	16	1.689e-08
9	7.119e-03	17	2.135e-09
10	1.356e-03	18	2.549e-10
11	2.415e-04	19	2.893e-11
12	3.990e-05	20	3.496e-12

An exponential convergence rate can be observed from the above table. For comparison, we also solve Example 2.4.1 using the finite-difference method. We use the central differences for the derivatives:

$$u'' \approx \frac{U_{j+1} - 2U_j + U_{j-1}}{h^2}, \qquad u' \approx \frac{U_{j+1} - U_{j-1}}{2h}, \quad h = \frac{2}{N}.$$

As usual the mesh points are given by $x_j = -1 + jh$. The maximum errors given by the finite-difference method are listed below:

N	Maximum error	N	Maximum error
16	3.100e+00	128	4.968e-02
32	7.898e-01	256	1.242e-02
64	1.984e-01	512	3.106e-03

As expected, the convergence rate for the central difference method is 2. The error obtained by the finite differences with $N = 512$ is almost the same as that obtained by the spectral method with $N = 10$.

The following example deals with BVPs with the general boundary conditions. We follow `CODE PSBVP.3` and use MATLAB to get the following results.

Example 2.4.2 *Consider the same problem as above, except with different boundary conditions*:

$$u(-1)-u'(-1) = -4e^{-5}+\sin(1)+2\cos(1), \quad u(1)+u'(1) = 6e^{5}+\sin(1)+2\cos(1).$$

The exact solution is also $u(x) = e^{5x} + \sin(x^2)$.

The numerical results are given below:

N	Maximum error	N	Maximum error
5	3.269e+01	13	3.254e-04
6	9.696e+00	14	4.903e-05
7	2.959e+00	15	8.823e-06
8	7.292e-01	16	1.164e-06
9	1.941e-01	17	1.884e-07
10	3.996e-02	18	2.204e-08
11	9.219e-03	19	3.225e-09
12	1.609e-03	20	3.432e-10

It is observed that the convergence rate for problems with general boundary conditions is slower than that for problems with Dirichlet boundary conditions.

Exercise 2.4

Problem 1 Consider the following problem with one boundary layer,

$$\epsilon U'' + \frac{1}{2}U'(x) = 0, \qquad x \in (-1, 1),$$

with $U(-1) = 0$ and $U(1) = 1$. This problem has the exact solution

$$U(x) = \left(1 - e^{-(x+1)/2\epsilon}\right)\left(1 - e^{-1/\epsilon}\right)^{-1}.$$

(1) Solve this problem for $\epsilon = 10^{-3}$ with $N = 64, 128$ and $\epsilon = 10^{-4}$ with $N = 128, 256$.

(2) Calculate the L^1-error, $\sum_{j=1}^{N-1} |u^N(x_j) - U(x_j)|/(N-1)$, and also plot the point-wise errors.

Problem 2 Use the Chebyshev spectral method to solve the nonlinear Poisson-Boltzmann equation[165]:

$$u_{xx} = e^u, \qquad -1 < x < 1, \qquad u(-1) = u(1) = 0. \tag{2.4.15}$$

(`Hint:` This is a nonlinear problem. A simple iterative method can be used to solve the resulting nonlinear system. Namely, solve $\tilde{D}^2\vec{v}_{\rm new} = \exp(\vec{v}_{\rm old}$, where $\tilde{D}^2$ is an $(N-1)\times(N-1)$ matrix obtained by stripping D^2 of its first and last rows and columns. In MATLAB notation: $\tilde{D}^2 = D^2(1:N-1, 1:N-1)$.)

2.5 Collocation method in the weak form and preconditioning

Collocation methods in the weak form
Finite difference preconditioning
Finite element preconditioning

The collocation method presented in Section 2.4 is derived by asking that the approximation solution satisfy *exactly* the boundary conditions and the equation at the interior collocation points. Alternatively, we can also define an approximate solution through a variational formulation which is more suitable for error analysis and for designing effective preconditioners.

Collocation methods in the weak form

A variational method usually preserves essential properties of the continuous problem such as coercivity, continuity and symmetry of the bilinear form, and leads to optimal error estimates.

Consider (2.4.1) and (2.4.2). Without loss of generality, we shall assume $c_\pm = 0$. We introduce

$$H^1_\star(I) = \{v \in H^1(I) : u(-1) = 0 \text{ if } b_- = 0;\ u(1) = 0 \text{ if } b_+ = 0\}, \qquad (2.5.1)$$

and

$$h_- = \begin{cases} 0 & \text{if } a_-b_- = 0, \\ a_-/b_- & \text{if } a_-b_- \neq 0, \end{cases} \qquad h_+ = \begin{cases} 0 & \text{if } a_+b_+ = 0, \\ a_+/b_+ & \text{if } a_+b_+ \neq 0. \end{cases} \qquad (2.5.2)$$

Then, the *Galerkin method with numerical integration* for (2.4.1) and (2.4.2) with $c_\pm = 0$ is: Find $u_N \in X_N = P_N \cap H^1_\star(I)$ such that

$$b_N(u_N, v_N) = \langle f, v\rangle_N, \qquad \forall v_N \in X_N, \qquad (2.5.3)$$

where

$$b_N(u_N, v_N) := \epsilon\langle u_N', v_N'\rangle_N + \epsilon h_+ u_N(1)v_N(1) - \epsilon h_- u_N(-1)v_N(-1)$$

$$+ \langle p(x)u'_N, v_N\rangle_N + \langle q(x)u_N, v_N\rangle_N,$$

with $\langle\cdot,\cdot\rangle_N$ denoting the discrete inner product associated with the Legendre-Gauss-Lobatto quadrature. We note that an essential difficulty appears at the boundaries with mixed boundary conditions if we want to use the Chebyshev-Gauss-Lobatto quadrature. This difficulty can be overcome by replacing X_N by $\tilde{X}_N = \{u \in P_N : a_\pm u(\pm 1) + b_\pm u(\pm 1) = 0\}$, see Section 3.1.

We now attempt to re-interpret (2.5.3) into a collocation form. To fix the idea, we assume $b_\pm \neq 0$ and denote

$$\begin{aligned}
&u_N(x) = \sum_{k=0}^{N} u_N(x_k)h_k(x), && \bar{w} = (u_N(x_0), u_N(x_1), \cdots, u_N(x_N))^{\mathrm{T}},\\
&a_{kj} = b_N(h_j, h_k), && A = (a_{kj})_{k,j=0}^{N},\\
&\bar{f} = (f(x_0), f(x_1), \cdots, f(x_N))^{\mathrm{T}}, && W = \mathrm{diag}(\omega_0, \omega_1, \cdots, \omega_N),
\end{aligned}$$

where $\{\omega_k\}_{k=0}^{N}$ are the weights in the Legendre-Gauss-Lobatto quadrature. Then, (2.5.3) is equivalent to the linear system

$$A\bar{w} = W\bar{f}. \tag{2.5.4}$$

The entries a_{kj} can be determined as follows. Using (1.2.22) and integration by parts, we have

$$\begin{aligned}
\langle h'_j, h'_k\rangle_N = (h'_j, h'_k) &= -(h''_j, h_k) + h'_j h_k|_{\pm 1}\\
&= -(D^2)_{kj}\omega_k + d_{0j}\delta_{0k} - d_{Nj}\delta_{Nk}.
\end{aligned} \tag{2.5.5}$$

Consequently,

$$\begin{aligned}
a_{kj} =&[-\epsilon(D^2)_{kj} + p(x_k)d_{kj} + q(x_k)\delta_{kj}]\omega_k + \epsilon(d_{0j} + h_+\delta_{0j})\delta_{0k}\\
&- \epsilon(d_{Nj} + h_-\delta_{Nj})\delta_{Nk}, \qquad 0 \leqslant k, j \leqslant N.
\end{aligned} \tag{2.5.6}$$

Note that here the matrix A is of order $(N+1)\times(N+1)$, instead of order $(N-1)\times(N-1)$ as in the pure collocation case. We observe that

$$\langle u'_N, h'_k\rangle_N = -u''_N(x_k)\omega_k + u'_N(1)\delta_{0k} - u'_N(-1)\delta_{Nk}.$$

Thus, taking $v_N = h_j(x)$ in (2.5.3) for $j = 0, 1, \cdots, N$, and observing that $\omega_0 = \omega_N = 2/N(N+1)$, we find

$$-\epsilon u''_N(x_j) + p(x_j)u'_N(x_j) + q(x_j)u_N(x_j) = f(x_j), \quad 1 \leqslant j \leqslant N-1,$$

$$a_\pm u_N(\pm 1) + b_\pm u'_N(\pm 1) = \frac{b_\pm}{\epsilon}\frac{2}{N(N+1)}\tau_\pm, \tag{2.5.7}$$

where

$$\tau_\pm = f(\pm 1) - \{-\epsilon u''_N(\pm 1) + p(\pm 1)u'_N(\pm 1) + q(\pm 1)u_N(\pm 1)\}.$$

We see that the solution of (2.5.3) satisfies (2.4.1) exactly at the interior collocation points $\{x_j\}_{j=1}^{N-1}$, but the boundary condition (2.4.2) (with $c_\pm = 0$) is only satisfied approximately with an error proportional to the residue of the equation (2.4.1), with u replaced by the approximate solution u_N, at the boundary. Thus, (2.5.3) does not correspond exactly to a collocation method and is referred to as *collocation method in the weak form*. We note however that in the Dirichlet case (i.e. $b_\pm = 0$), the collocation method in the weak form (2.5.7) is *equivalent* to the usual *collocation method*.

The collocation methods, either in the strong form or weak form, lead to a full and ill-conditioned linear system. Hence, a direct solution method such as Gaussian elimination is only feasible for one-dimensional problems with a small to moderate number of unknowns. For multi-dimensional problems and/or problems with large number of unknowns, an iterative method with an appropriate preconditioner should be used. To this end, it is preferable to first transform the problem (2.4.1) and (2.4.2) into a self-adjoint form. We observe first that without loss of generality we may assume $c_\pm = 0$ by modifying the right-hand side function f. Then, multiplying the function

$$a(x) = \exp\left(-\frac{1}{\epsilon}\int p(x)\mathrm{d}x\right) \tag{2.5.8}$$

to (2.4.1) and noting that $-\epsilon a'(x) = a(x)p(x)$, we find that (2.4.1) and (2.4.2) with $c_\pm = 0$ can be written as

$$\begin{aligned} &-(a(x)u'(x))' + b(x)u = g(x), \quad x \in (-1,1), \\ &a_-u(-1) + b_-u'(-1) = 0, \quad a_+u(1) + b_+u'(1) = 0, \end{aligned} \tag{2.5.9}$$

where $b(x) = a(x)q(x)/\epsilon$ and $g(x) = a(x)f(x)/\epsilon$.

Finite difference preconditioning

The collocation method in the strong form for (2.5.9) is: Find $u_N \in P_N$ such that

$$\begin{aligned} &-(au'_N)'(x_j) + b(x_j)u_N(x_j) = g(x_j), \quad 1 \leqslant j \leqslant N-1, \\ &a_-u_N(-1) + b_-u'_N(-1) = 0, \quad a_+u_N(1) + b_+u'_N(1) = 0. \end{aligned} \tag{2.5.10}$$

As demonstrated earlier, (2.5.10) can be rewritten as an $(N-1)\times(N-1)$ linear system

$$A\bar{w}=\bar{f}, \tag{2.5.11}$$

where the unknowns are $\{w_j = u_N(x_j)\}_{j=1}^{N-1}$, $\bar{w} = (w_1,\cdots,w_{N-1})^{\mathrm{T}}$ and $\bar{f} = (f(x_1),\cdots,f(x_{N-1}))^{\mathrm{T}}$. The entries of A are given in Section 2.1.

As suggested by Orszag [125], we can build a preconditioner for A by using a finite difference approximation to (2.5.9). Let us define

$$\begin{aligned} &h_k = x_{k-1}-x_k, \qquad \tilde{h}_k = \frac{1}{2}(x_{k-1}-x_{k+1}),\\ &x_{k+\frac{1}{2}} = \frac{1}{2}(x_{k+1}+x_k), \quad a_{k+\frac{1}{2}} = a(x_{k+\frac{1}{2}}). \end{aligned} \tag{2.5.12}$$

Then, the second-order finite difference scheme for (2.5.9) with first-order one-sided difference at the boundaries reads:

$$\begin{aligned} &-\frac{a_{i-\frac{1}{2}}}{\tilde{h}_i h_i}w_{i-1}+\left(\frac{a_{i-\frac{1}{2}}}{\tilde{h}_i h_i}+\frac{a_{i+\frac{1}{2}}}{\tilde{h}_i h_{i+1}}\right)w_i\\ &-\frac{a_{i+\frac{1}{2}}}{\tilde{h}_i h_{i+1}}w_{i+1}+b(x_i)w_i = g(x_i), \quad 1\leqslant i\leqslant N-1,\\ &a_- w_N + b_-(w_{N-1}-w_N)/h_N = 0, \qquad a_+ w_0 + b_+(w_0-w_1)/h_1 = 0. \end{aligned} \tag{2.5.13}$$

We can rewrite (2.5.13) in the matrix form

$$A_{fd}\bar{w}=\bar{f}, \tag{2.5.14}$$

where A_{fd} is a non-symmetric tridiagonal matrix. It has been shown (cf. [125], [80], [91]) that in the Dirichlet case, A_{fd}^{-1} is an optimal preconditioner for A, but $\mathrm{cond}(A_{fd}^{-1}A)$ deteriorates with other boundary conditions. The main reason for this deterioration is that the collocation method in the strong form with non-Dirichlet boundary conditions cannot be cast into a variational formulation.

Remark 2.5.1 The above discussion is valid for both the Legendre and Chebyshev collocation methods.

Finite element preconditioning

A more robust preconditioner can be constructed by using a finite element approximation, which is always based on a variational formulation. Thus, it can only be used for the preconditioning of collocation methods which can be cast into a variational formulation. Namely, the collocation method for the Dirichlet boundary condi-

tions or the collocation method in the weak form for the general boundary conditions.

We consider first the treatment of the general boundary conditions. Let us denote

$$X_h = \{u \in H^1_\star(I) : u|_{[x_{i+1},x_i]} \in P_1, i = 0, 1, \cdots, N-1\}. \tag{2.5.15}$$

Then,the piecewise linear finite element approximation to (2.5.9) is: Find $u_h \in X_h$ such that for all $v_h \in X_h$,

$$b_h(u_h, v_h) = \langle f, v_h\rangle_h, \tag{2.5.16}$$

where

$$\begin{aligned} b_h(u_h, v_h) :=& \langle a u_h', v_h'\rangle_h + a(1)h_+ u_h(1)v_h(1) \\ & - a(-1)h_- u_h(-1)v_h(-1) + \langle b u_h, v_h\rangle_h, \end{aligned}$$

and $\langle\cdot,\cdot\rangle_h$ is an appropriate discrete inner product associated with the piecewise linear finite element approximation.

To demonstrate the idea, we assume $b_\pm \neq 0$. Let us denote for $k = 1, \cdots, N-1$,

$$\hat{h}_k(x) = \begin{cases} \dfrac{x - x_{k+1}}{x_k - x_{k+1}}, & x \in [x_{k+1}, x_k], \\ \dfrac{x_{k-1} - x}{x_{k-1} - x_k}, & x \in [x_k, x_{k-1}], \\ 0, & \text{otherwise;} \end{cases} \tag{2.5.17}$$

$$\hat{h}_0(x) = \begin{cases} \dfrac{x - x_1}{x_0 - x_1}, & x \in [x_1, x_0], \\ 0, & \text{otherwise;} \end{cases} \tag{2.5.18}$$

$$\hat{h}_N(x) = \begin{cases} \dfrac{x_{N-1} - x}{x_{N-1} - x_N}, & x \in [x_N, x_{N-1}], \\ 0, & \text{otherwise.} \end{cases} \tag{2.5.19}$$

It can be verified that $X_h = \text{span}\{\hat{h}_0, \hat{h}_1, \cdots, \hat{h}_N\}$. We further set

$$u_h(x) = \sum_{k=0}^{N} u_h(x_k)\hat{h}_k(x), \quad w = (u_h(x_0), \cdots, u_h(x_N))^{\mathrm{T}},$$

$$b_{kj} = b_h(\hat{h}_j, \hat{h}_k), \quad B_{fe} = (b_{kj})_{k,j=0}^{N}, \quad m_{kj} = \langle \hat{h}_j, \hat{h}_k\rangle_h,$$

$$M_{fe} = (m_{kj})_{k,j=0}^{N}, \quad f = (f(x_0), \cdots, f(x_N))^{\mathrm{T}}.$$

Then, (2.5.16) is equivalent to the linear system

$$B_{fe}\bar{w} = M_{fe}\bar{f} \quad \text{or} \quad M_{fe}^{-1}B_{fe}\bar{w} = \bar{f}. \tag{2.5.20}$$

On the other hand, as demonstrated earlier, we can formulate the linear system associated with the Legendre-collocation method for (2.4.5)–(2.5.9) in the weak form

$$A\bar{w} = W\bar{f} \quad \text{or} \quad W^{-1}A\bar{w} = \bar{f}. \tag{2.5.21}$$

Since both (2.5.21) and (2.5.20) provide approximate solutions to (2.5.9), it is known (cf. [127]) that $M_{fe}^{-1}B_{fe}$ is a good preconditioner for $W^{-1}A$.

Exercise 2.5

Problem 1 Consider the problem

$$-u_{xx} + u = f, \qquad u(-1) = 0, \qquad u'(1) = 0.$$

Compute the condition number of the preconditioner matrix $B_{fe}^{-1}M_{fe}W^{-1}A$ described above for $N = 8, 16, 32, 64$.

Problem 2 Solve the Poisson-Boltzmann equation described in Problem 2 of Section 2.4 by using a preconditioned iterative method using a finite element preconditioner.

Chapter 3

Spectral-Galerkin Methods

Contents

3.1 General setup 105
3.2 Legendre-Galerkin method 109
3.3 Chebyshev-Galerkin method 114
3.4 Chebyshev-Legendre Galerkin method 118
3.5 Preconditioned iterative method 121
3.6 Spectral-Galerkin methods for higher-order equations 126
3.7 Error estimates 131

An alternative approach to spectral-collocation is the so called spectral-Galerkin method which is based on a variational formulation and uses, instead of Lagrange polynomials, compact combinations of orthogonal polynomials as basis functions. It will be shown that by choosing proper basis functions, the spectral-Galerkin method may lead to well conditioned linear systems with sparse matrices for problems with constant or polynomial coefficients. In this chapter, we present the Legendre- and Chebyshev-Galerkin algorithms and their error analysis for a class of one-dimensional problems.

3.1 General setup

Reformulation of the problem
(Weighted) Galerkin formulation

We will demonstrate the ideas of spectral-Galerkin methods for the two-point boundary-value problem:

$$-\epsilon U'' + p(x)U' + q(x)U = F, \quad x \in I = (-1, 1), \tag{3.1.1}$$

with the general boundary condition

$$a_- U(-1) + b_- U'(-1) = c_-, \quad a_+ U(1) + b_+ U'(1) = c_+. \tag{3.1.2}$$

This includes in particular the Dirichlet ($a_\pm = 1$ and $b_\pm = 0$), the Neumann ($a_\pm = 0$ and $b_\pm = 1$), and the mixed ($a_- = b_+ = 0$ or $a_+ = b_- = 0$) boundary conditions. Whenever possible, we will give a uniform treatment for all these boundary conditions. We assume that $a_\pm, b_\pm$ and $c_\pm$ satisfy (2.4.3) so that the problem (3.1.1) and (3.1.2) is well-posed.

Unlike the pseudospectral or collocation methods which require the approximate solution to satisfy (3.1.1), the Galerkin method is based on variational formulation. Hence, it is desirable, whenever possible, to reformulate the problem (3.1.1) and (3.1.2) into a self-adjoint form.

Reformulation of the problem

Let us first reduce the problem (3.1.1) and (3.1.2) to a problem with homogeneous boundary conditions:

• `Case 1` $a_\pm = 0$ `and` $b_\pm \neq 0$`.` We set $\tilde{u} = \beta x^2 + \gamma x$, where β and γ are uniquely determined by asking $\tilde{u}$ to satisfy (3.1.2), namely,

$$-2b_-\beta + b_-\gamma = c_-, \quad 2b_+\beta + b_+\gamma = c_+. \tag{3.1.3}$$

• `Case 2` $a_-^2 + a_+^2 \neq 0$`.` We set $\tilde{u} = \beta x + \gamma$, where β and γ can again be uniquely determined by asking that $\tilde{u}$ to satisfy (3.1.2). Indeed, we have

$$(-a_- + b_-)\beta + a_-\gamma = c_-, \quad (a_+ + b_+)\beta + a_+\gamma = c_+, \tag{3.1.4}$$

whose determinant is

$$\mathsf{DET} = -a_-a_+ + b_-a_+ - a_-a_+ - b_+a_-.$$

Thus, (2.4.3) implies that $b_- \leqslant 0$ and $b_+ \geqslant 0$ which imply that $\mathsf{DET} < 0$.

We now set $u = U - \tilde{u}$ and $f = F - (-\epsilon\tilde{u}'' + p(x)\tilde{u}' + q(x)\tilde{u})$. Then u satisfies the equation

$$-\epsilon u'' + p(x)u' + q(x)u = f, \quad \text{in } I = (-1, 1), \tag{3.1.5}$$

with the homogeneous boundary conditions

$$a_- u(-1) + b_- u'(-1) = 0, \quad a_+ u(1) + b_+ u'(1) = 0. \tag{3.1.6}$$

Next, we transform the above equation into a self-adjoint form which is more suitable for error analysis and for developing efficient numerical schemes. To this end, multiplying the function (2.5.8)–(3.1.5) and noting $-\epsilon d'(x) = a(x)p(x)$, we find that (3.1.5) is equivalent to

$$-(a(x)u'(x))' + b(x)u = g(x), \tag{3.1.7}$$

where $b(x) = a(x)q(x)/\epsilon$ and $g(x) = a(x)f(x)/\epsilon$.

(Weighted) Galerkin formulation

We shall look for approximate solutions of (3.17) and (3.16) in the space

$$X_N = \{v \in P_N : \ a_\pm v(\pm 1) + b_\pm v'(\pm 1) = 0\}. \tag{3.1.8}$$

Note that we require the approximate solution satisfies the **exact** boundary conditions. This is different from a usual finite element approach where only the Dirichlet boundary conditions are enforced while the general boundary conditions (3.1.6) are treated as natural boundary conditions. The main advantage of our approach is that it leads to sparse matrices for problems with constant or polynomial coefficients (see the next two sections), while the disadvantage is that a stronger regularity on the solution is required for convergence.

Let $\omega(x)$ be a positive weight function and $I_N : C(-1, 1) \to P_N$ be the interpolating operator associated with Gauss-Lobatto points. Then, the (weighted) spectral-Galerkin method for (3.17) and (3.16) is to look for $u_N \in X_N$ such that

$$-([I_N(a(x)u_N')]', v_N)_\omega + (I_N(b(x)u_N), v_N)_\omega = (I_N f, v_N)_\omega \qquad \forall\, v_N \in X_N, \tag{3.1.9}$$

Remark 3.1.1 We note that (3.1.9) is actually a hybrid of a Galerkin and a pseudospectral method since a pure Galerkin method would not use any interpolation operator in (3.1.9). However, since, for example, the integral $\int_I f v_N \, \mathrm{d}x$ cannot be computed exactly so f, and other products of two functions, are always replaced by

their interpolants in practical computations. We shall take this approach throughout this book and still call it a Galerkin method.

Given a set of basis functions $\{\phi_k\}_{k=0,1,\cdots,N-2}$ for X_N, we define

$$\begin{aligned} &f_k = (I_N f, \phi_k)_\omega, \quad \bar{f} = (f_0, f_1, \cdots, f_{N-2})^{\mathrm{T}}; \\ &u_N(x) = \sum_{n=0}^{N-2} \hat{u}_n \phi_n(x), \quad \bar{u} = (\hat{u}_0, \hat{u}_1, \cdots, \hat{u}_{N-2})^{\mathrm{T}}, \\ &s_{kj} = -([I_N(a(x)\phi_j')]', \phi_k)_\omega, \quad m_{kj} = (I_N(b(x)\phi_j), \phi_k)_\omega. \end{aligned} \tag{3.1.10}$$

Hence, the stiffness and mass matrices are

$$S = (s_{kj})_{0\leqslant k,j\leqslant N-2}, \quad M = (m_{kj})_{0\leqslant k,j\leqslant N-2}. \tag{3.1.11}$$

By setting $u_N(x) = \sum_{n=0}^{N-2} \hat{u}_n \phi_n(x)$ and $v_N(x) = \phi_j(x)$, $0 \leqslant j \leqslant N-2$, in (3.1.9), we find that the equation (3.1.9) is equivalent to the linear system

$$(S + M)\bar{u} = \bar{f}. \tag{3.1.12}$$

Unfortunately, for problems with variable coefficients $a(x)$ and $b(x)$, S and M are usually full matrices, and it is very costly to compute them and to solve (3.1.12). However, as we shall demonstrate in the next two sections, S and M will be sparse (or have very special structures) for problems with constant coefficients. Then, in Section 3.5, we shall show how to use preconditioned iterative approach to solve (3.1.12) with variable coefficients.

Exercise 3.1

Problem 1 Let $H^1_\star(I)$ and $h_\pm$ be defined as in (2.5.1) and (2.5.2). Then, the usual variational formulation for (3.1.7) with (3.1.6) is: Find $u \in H^1_\star(I)$ such that

$$\begin{aligned} &(au', v') + a(1)h_+u(1)v(1) - a(-1)h_-u(-1)v(-1) + (bu, v) \\ =&(g, v), \qquad \forall v \in H^1_\star(I). \end{aligned} \tag{3.1.13}$$

1. Show that a sufficiently smooth solution of (3.1.13) is a classical solution of (3.1.7) with (3.2.2).

2. Let $X_N = P_N \cap H^1_\star(I)$. Write down the (non-weighted) Galerkin approximation in X_N for (3.1.13) and determine the corresponding linear system as in (3.1.12)

with (3.1.10) and (3.1.11).

3. Attempt to construct a weighted Galerkin approximation in X_N to (3.1.13) and explain the difficulties.

3.2 Legendre-Galerkin method

Basis functions, stiffness and mass matrices
Algorithm

To illustrate the essential features of the spectral-Galerkin methods, we shall consider, here and in the next two sections, the model problem

$$-u'' + \alpha u = f, \qquad \text{in} I = (-1, 1), \tag{3.2.1}$$

$$a_{\pm} u(\pm 1) + b_{\pm} u'(\pm 1) = 0. \tag{3.2.2}$$

We assume that α is a non-negative constant. Extension to more general problems (2.4.1) and (2.4.2) will be addressed in Section 3.5.

In this case, the spectral Galerkin method becomes: Find $u_N \in X_N$ such that

$$\int_I u_N' v_N' \mathrm{d}x + \alpha \int_I u_N \, v_N \, \mathrm{d}x = \int_I I_N f \, v_N \, \mathrm{d}x, \quad \forall \, v_N \in X_N, \tag{3.2.3}$$

which we refer to as the Legendre-Galerkin method for (3.2.1) and (3.2.2).

Basis functions, stiffness and mass matrices

The actual linear system for (3.2.3) will depend on the basis functions of X_N. Just as in the finite-element methods, neighboring points are used to form basis functions so as to minimize their interactions in the physical space, neighboring orthogonal polynomials should be used to form basis functions in a spectral-Galerkin method so as to minimize their interactions in the frequency space. Therefore, we look for basis functions of the form

$$\phi_k(x) = L_k(x) + a_k L_{k+1}(x) + b_k L_{k+2}(x). \tag{3.2.4}$$

Lemma 3.2.1 *For all $k \geqslant 0$, there exist unique $\{a_k, b_k\}$ such that $\phi_k(x)$ of the form* (3.2.4) *satisfies the boundary condition* (3.2.2).

Proof Since $L_k(\pm 1) = (\pm 1)^k$ and $L_k'(\pm 1) = \frac{1}{2}(\pm 1)^{k-1} k(k+1)$, the boundary

condition (3.2.2) leads to the following system for $\{a_k, b_k\}$:

$$\begin{aligned} &\{a_+ + b_+(k+1)(k+2)/2\}a_k + \{a_+ + b_+(k+2)(k+3)/2\}b_k \\ =& -a_+ - b_+k(k+1)/2, \\ &-\{a_- - b_-(k+1)(k+2)/2\}a_k + \{a_- - b_-(k+2)(k+3)/2\}b_k \\ =& -a_- + b_-k(k+1)/2. \end{aligned} \tag{3.2.5}$$

The determinant of the above system is

$$\mathsf{DET}_k = 2a_+a_- + a_-b_+(k+2)^2 - a_+b_-(k+2)^2 - b_-b_+(k+1)(k+2)^2(k+3)/2.$$

We then derive from (2.4.3) that the four terms (including the signs before them) of DET_k are all positive for any k. Hence, $\{a_k, b_k\}$ can be uniquely determined from (3.2.5), namely:

$$\begin{aligned} a_k =& -\Big\{\Big(a_+ + \frac{b_+}{2}(k+2)(k+3)\Big)\Big(-a_- + \frac{b_-}{2}k(k+1)\Big) \\ &-\Big(a_- - \frac{b_-}{2}(k+2)(k+3)\Big)\Big(-a_+ - \frac{b_+}{2}k(k+1)\Big)\Big\}\Big/\mathsf{DET}_k, \\ b_k =& \Big\{\Big(a_+ + \frac{b_+}{2}(k+1)(k+2)\Big)\Big(-a_- + \frac{b_-}{2}k(k+1)\Big) \\ &+\Big(a_- - \frac{b_-}{2}(k+1)(k+2)\Big)\Big(-a_+ - \frac{b_+}{2}k(k+1)\Big)\Big\}\Big/\mathsf{DET}_k. \end{aligned}$$

This completes the proof of this lemma. □

Remark 3.2.1 We note in particular that

• if $a_\pm = 1$ and $b_\pm = 0$ (Dirichlet boundary conditions), we have $a_k = 0$ and $b_k = -1$. Hence, we find from (1.4.12) that

$$\phi_k(x) = L_k(x) - L_{k+2}(x) = \frac{2k+3}{2(k+1)}J_{k+2}^{-1,-1}(x).$$

• if $a_\pm = 0$ and $b_\pm = 1$ (Neumann boundary conditions), we have $a_k = 0$ and $b_k = -k(k+1)/((k+2)(k+3))$.

It is obvious that $\{\phi_k(x)\}$ are linearly independent. Therefore, by a dimension argument we have

$$X_N = \mathrm{span}\{\phi_k(x) :\ k = 0, 1, \cdots, N-2\}.$$

Remark 3.2.2 In the very special case $-u_{xx} = f$, $u_x(\pm 1) = 0$, with the condition

$\int_{-1}^{1} f\mathrm{d}x = 0$, since the solution is only determined up to a constant, we should use

$$X_N = \mathrm{span}\{\phi_k(x) :\ k = 1, \cdots, N-2\}.$$

This remark applies also to the Chebyshev-Galerkin method presented below.

Lemma 3.2.2 *The stiffness matrix S is a diagonal matrix with*

$$s_{kk} = -(4k+6)b_k, \qquad k = 0, 1, 2, \cdots. \tag{3.2.6}$$

The mass matrix M is a symmetric penta-diagonal matrix whose nonzero elements are

$$m_{jk} = m_{kj} = \begin{cases} \dfrac{2}{2k+1} + a_k^2 \dfrac{2}{2k+3} + b_k^2 \dfrac{2}{2k+5}, & j = k, \\ a_k \dfrac{2}{2k+3} + a_{k+1} b_k \dfrac{2}{2k+5}, & j = k+1, \\ b_k \dfrac{2}{2k+5}, & j = k+2. \end{cases} \tag{3.2.7}$$

Proof By integration by parts and taking into account the boundary condition (3.2.2), we find that

$$\begin{aligned} s_{jk} &= -\int_I \phi_k''(x)\, \phi_j(x)\mathrm{d}x \\ &= \int_I \phi_k'(x)\, \phi_j'(x)\mathrm{d}x + \frac{a_+}{b_+}\phi_k(1)\phi_j(1) - \frac{a_-}{b_-}\phi_k(-1)\phi_j(-1) \\ &= -\int_I \phi_k(x)\, \phi_j''(x)\mathrm{d}x = s_{kj}, \end{aligned} \tag{3.2.8}$$

where a_+/b_+ (resp. a_-/b_-) should be replaced by zero when $b_+ = 0$ (resp. $b_- = 0$). It is then obvious from (3.2.8) and the definition of $\{\phi_k(x)\}$ that S is a diagonal matrix. Thanks to (1.3.22e) and (1.3.19), we find

$$\begin{aligned} s_{kk} &= -b_k \int_I L_{k+2}''(x) L_k(x)\mathrm{d}x \\ &= -b_k \left(k + \frac{1}{2}\right)(4k+6) \int_I L_k^2 \mathrm{d}x = -b_k(4k+6). \end{aligned}$$

The nonzero entries for M can be easily obtained using (1.3.19). □

Remark 3.2.3 An immediate consequence is that $\{\phi_k\}_{k=0}^{N-2}$ form an orthogonal basis in X_N with respect to the inner product $-(u_N'', v_N)$. Furthermore, an *orthonormal* basis of X_N is given by $\{\tilde{\phi}_k := -\frac{1}{b_k(4k+6)}\phi_k\}_{k=0}^{N-2}$.

Algorithm

Hence, by setting $u_N = \sum_{k=0}^{N-2} \tilde{u}_k\phi_k$, $\bar{u} = (\tilde{u}_0, \tilde{u}_1, \cdots, \tilde{u}_{N-2})^{\mathrm{T}}$, and

$$\tilde{f}_k = (I_N f, \psi_k), \quad \bar{f} = (\tilde{f}_0, \tilde{f}_1, \cdots, \tilde{f}_{N-2})^{\mathrm{T}}, \tag{3.2.9}$$

the linear system (3.2.3) becomes

$$(\alpha M + S)\bar{u} = \bar{f}, \tag{3.2.10}$$

where M and S are $(N-2)\times(N-2)$ matrices with entries m_{ij} and s_{ij}, respectively.

In summary: given the values of f at LGL points $\{x_i\}_{0\leqslant i\leqslant N}$, we determine the values of u_N, solution of (3.2.3), at these LGL points as follows:

1. (Pre-computation) Compute LGL points, $\{a_k, b_k\}$ and nonzero elements of S and M;
2. Evaluate the Legendre coefficients of $I_N f(x)$ from $\{f(x_i)\}_{i=0}^N$ (backward Legendre transform) and evaluate $\bar{f}$ in (3.2.9);
3. Solve $\bar{u}$ from (3.2.10);
4. Determine $\{\hat{u}_j\}_{j=0}^N$ such that $\sum_{j=0}^{N-2}\tilde{u}_j\phi_j(x) = \sum_{j=0}^N \hat{u}_j L_j(x)$;
5. Evaluate $u_N(x_j) = \sum_{i=0}^N \hat{u}_i L_i(x_j)$, $j = 0, 1, \cdots, N$ (forward Legendre transform).

A pseudo-code outlines the above solution procedure is provided below:

```
CODE LG-PSN-1D
Input N, collocation points x_k and f(x_k) for k = 0,1,...,N
Compute a_k, b_k, s_kk, m_kj
%Backward Legendre transform
for k=0 to N-1 do
    g_k = (2k+1)/(N(N+1)) Σ_{j=0}^{N} f(x_j) L_k(x_j)/L_N(x_j)^2
endfor
g_N = 1/(N+1) Σ_{j=0}^{N} f(x_j) 1/L_N(x_j)
%Evaluate f̄ from  f_k = (Σ_{j=0}^{N} g_j L_j(x), φ_k(x))
for k=0 to N-2 do
    f_k = g_k/(k+1/2) + a_k g_{k+1}/(k+3/2) + b_k g_{k+2}/(k+5/2)
endfor
Solve (S+αM)ū = f̄
```

```
%Evaluate g_k from sum_{j=0}^{N-2} û_j φ_j(x) = sum_{j=0}^{N} g_j L_j(x)
g_0 = û_0,   g_1 = û_1 + a_0 û_0
for k=2 to N-2 do
    g_k = û_k + a_{k-1} û_{k-1} + b_{k-2} û_{k-2}
endfor
g_{N-1} = a_{N-2} û_{N-2} + b_{N-3} û_{N-3},  g_N = b_{N-2} û_{N-2}
%forward Legendre transform
for k=0 to N do
    û_k = sum_{j=0}^{N} g_j L_j(x_k)
endfor
Output û_0, û_1, ..., û_N
```

Although the solution of the linear system (3.1.12) can be found in $\mathcal{O}(N)$ flops, the two discrete Legendre transforms in the above procedure cost about $2N^2$ flops. To reduce the cost of the discrete transforms between physical and spectral spaces, a natural choice is to use Chebyshev polynomials so that the discrete Chebyshev transforms can be accelerated by using FFT.

Exercise 3.2

Problem 1 Continue with the Problem 1 in Section 3.1. Let $a_{\pm} = 0$ and take $a(x) = b(x) \equiv 1$. Construct a set of basis functions for X_N and derive the corresponding matrix system. Compare with the Legendre-Galerkin method in this section.

Problem 2 Consider the problem

$$u - u_{xx} = f; \;\; u(-1) = 0, \; u(1) = 1,$$

with the exact solution:

$$u(x) = \begin{cases} 0, & x \in [-1, 0], \\ x^{\gamma}, & x \in (0, 1], \end{cases}$$

where $\gamma = 4, \; 5, \; 6$, and define

$$\|u - u_N\|^2_{N,\omega} = \langle u - u_N, u - u_N \rangle_{N,\omega} = \sum_{i=0}^{N} (u - u_N)^2(x_i)\omega_i \, ,$$

where $\{x_i\}$ are the (Legendre or Chebyshev) Gauss-Lobatto points and $\{\omega_i\}$ are the associated weights.

Solve the above problem using the Legendre-Galerkin method. Take $N = 2^i$

with $i = 4, 5, 6, 7, 8, 9$. Plot $\log_{10} \|u - u_N\|_N / \log_{10} N$ for each γ. Explain your results.

3.3 Chebyshev-Galerkin method

Basis function, stiffness and mass matrices
Algorithm

We set $\omega(x) = (1 - x^2)^{-\frac{1}{2}}$ and $f_N = I_N f$ which is the Chebyshev interpolation polynomial of f relative to the Chebyshev-Gauss-Lobatto points. Then (3.1.9) becomes

$$-\int_I u_N'' \, v_N \, \omega \mathrm{d}x + \alpha \int_I u_N \, v_N \, \omega(x) \mathrm{d}x = \int_I I_N f v_N \omega(x) \, \mathrm{d}x, \quad \forall \, v_N \in X_N, \tag{3.3.1}$$

which we refer to as the Chebyshev-Galerkin method for (3.2.1) and (3.2.2).

Basis functions, stiffness and mass matrices

As before, we would like to seek the basis functions of X_N of the form

$$\phi_k(x) = T_k(x) + a_k T_{k+1}(x) + b_k T_{k+2}(x). \tag{3.3.2}$$

Lemma 3.3.1 *Let us define*

$$\begin{aligned} a_k = & -\big\{(a_+ + b_+(k+2)^2)(-a_- + b_- k^2) \\ & -(a_- - b_-(k+2)^2)(-a_+ - b_+ k^2)\big\} / \mathsf{DET}_k, \\ b_k = & \big\{(a_+ + b_+(k+1)^2)(-a_- + b_- k^2) \\ & +(a_- - b_-(k+1)^2)(-a_+ - b_+ k^2)\big\} / \mathsf{DET}_k, \end{aligned} \tag{3.3.3}$$

with

$$\mathsf{DET}_k = 2a_+ a_- + (k+1)^2 (k+2)^2 (a_- b_+ - a_+ b_- - 2b_- b_+). \tag{3.3.4}$$

Then

$$\phi_k(x) = T_k(x) + a_k T_{k+1}(x) + b_k T_{k+2}(x) \tag{3.3.5}$$

satisfies the boundary condition (3.2.2)*.*

Proof Since $T_k(\pm 1) = (\pm 1)^k$ and $T_k'(\pm 1) = (\pm 1)^{k-1} k^2$, we find from (3.2.2) that

$\{a_k, b_k\}$ must satisfy the system

$$\begin{aligned}(a_+ + b_+(k+1)^2)a_k + (a_+ + b_+(k+2)^2)b_k &= -a_+ - b_+k^2,\\ -(a_- - b_-(k+1)^2)a_k + (a_- - b_-(k+2)^2)b_k &= -a_- + b_-k^2,\end{aligned} \tag{3.3.6}$$

whose determinant DET_k is given by (3.3.4). As in the Legendre case, the condition (2.4.3) implies that $\mathsf{DET}_k \neq 0$. Hence, $\{a_k, b_k\}$ are uniquely determined by (3.3.3). □

Therefore, we have by a dimension argument that $X_N = \mathrm{span}\{\phi_k(x) : k = 0, 1, \cdots, N-2\}$. One easily derives from (1.3.2) that the mass matrix M is a symmetric positive definite penta-diagonal matrix whose nonzero elements are

$$m_{jk} = m_{kj} = \begin{cases} c_k\dfrac{\pi}{2}(1 + a_k^2 + b_k^2), & j = k,\\ \dfrac{\pi}{2}(a_k + a_{k+1}b_k), & j = k+1,\\ \dfrac{\pi}{2}b_k, & j = k+2,\end{cases} \tag{3.3.7}$$

where $c_0 = 2$ and $c_k = 1$ for $k \geqslant 1$. However, the computation of s_{kj} is much more involved. Below, we derive explicit expressions of s_{kj} for two special cases.

Lemma 3.3.2 *For the case $a_\pm = 1$ and $b_\pm = 0$ (Dirichlet boundary conditions), we have $a_k = 0$, $b_k = -1$ and*

$$s_{kj} = \begin{cases} 2\pi(k+1)(k+2), & j = k\\ 4\pi(k+1), & j = k+2, k+4, k+6, \cdots\\ 0, & j < k \text{ or } j+k \text{ odd}\end{cases}. \tag{3.3.8}$$

For the case $a_\pm = 0$ and $b_\pm = 1$ (Neumann boundary conditions), we have $a_k = 0$, $b_k = -k^2/(k+2)^2$, and

$$s_{kj} = \begin{cases} 2\pi(k+1)k^2/(k+2), & j = k,\\ 4\pi j^2(k+1)/(k+2)^2, & j = k+2, k+4, k+6, \cdots,\\ 0, & j < k \text{ or } j+k \text{ odd.}\end{cases} \tag{3.3.9}$$

Proof One observes immediately that $s_{kj} = -\int_I \phi_j''\phi_k\omega\,\mathrm{d}x = 0$ for $j < k$. Hence, S is an upper triangular matrix. By the odd-even parity of the Chebyshev polynomials, we have also $s_{kj} = 0$ for $j + k$ odd.

Owing to (1.3.5), we have

$$\begin{aligned} T''_{k+2}(x) =& \frac{1}{c_k}(k+2)((k+2)^2-k^2)T_k(x) \\ &+ \frac{1}{c_{k-2}}(k+2)((k+2)^2-(k-2)^2)T_{k-2}(x)+\cdots \end{aligned} \tag{3.3.10}$$

We consider first the case $a_\pm = 1$ and $b_\pm = 0$. From (3.3.3), we find $\phi_k(x) = T_k(x) - T_{k+2}(x)$. It follows immediately from (3.3.10) and (1.3.2) that

$$\begin{aligned} &-(\phi''_k(x),\phi_k(x))_\omega = (T''_{k+2}(x),T_k(x))_\omega \\ =&(k+2)((k+2)^2-k^2)(T_k(x),T_k(x))_\omega = 2\pi(k+1)(k+2). \end{aligned} \tag{3.3.11}$$

Setting $\phi''_j(x) = \sum_{n=0}^{j} d_n T_n(x)$, we derive, by a simple computation using (3.3.10),

$$d_n = \begin{cases} 4(j+1)(j+2)/c_j, & n=j, \\ \{(j+2)^3 - j^3 - 2n^2\}/c_n, & n<j. \end{cases}$$

Hence for $j = k+2, k+4, \cdots$, we find

$$\begin{aligned} -(\phi''_j(x),\phi_k(x))_\omega &= d_k(T_k(x),T_k(x))_\omega - d_{k+2}(T_{k+2}(x),T_{k+2}(x))_\omega \\ &= 4\pi(k+1). \end{aligned}$$

The case $a_\pm = 0$ and $b_\pm = 1$ can be treated in a similar way. □

Algorithm

The Chebyshev-Galerkin method for (3.2.1) and (3.2.2) involves the following steps:

1. (Pre-computation) Compute $\{a_k, b_k\}$ and nonzero elements of S and M;
2. Evaluate the Chebyshev coefficients of $I_N f(x)$ from $\{f(x_i)\}_{i=0}^N$ (backward Chebyshev transform) and evaluate $\bar{f}$;
3. Solve $\bar{u}$ from (3.1.12);
4. Evaluate $u_N(x_j) = \sum_{i=0}^{N-2} \hat{u}_i \phi_i(x_j)$, $j = 0, 1, \cdots, N$ (forward Chebyshev transform).

Note that the forward and backward Chebyshev transforms can be performed by using the Fast Fourier Transform (FFT) in $\mathcal{O}(N \log_2 N)$ operations. However, the cost of Step 3 depends on the boundary conditions (3.2.2). For the special but important

cases described in the above Lemma, the special structure of S would allow us to solve the system (3.1.12) in $\mathcal{O}(N)$ operations. More precisely, in (3.3.8) and (3.3.9), the nonzero elements of S take the form $s_{kj} = a(j) * b(k)$, hence, a special Gaussian elimination procedure for (3.1.12) (cf. [139]) would only require $\mathcal{O}(N)$ flops instead of $\mathcal{O}(N^3)$ flops for a general full matrix.

Therefore, thanks to the FFT which can be used for the discrete Chebyshev transforms, the computational complexity of the Chebyshev-Galerkin method for the above cases is $\mathcal{O}(N \log N)$, which is quasi-optimal (i.e., optimal up to a logarithmic term).

The following pseudo-code outlines the solution procedure for (3.1.5) by the Chebyshev-Galerkin method:

CODE CG-PSN-1D
Input N
Set up collocation points x_k: x(j)=cos(πj/N), 0⩽j⩽N
Set up the coefficients $\tilde{c}_k$: $\tilde{c}$(0)=2, $\tilde{c}$(N)=2, $\tilde{c}$(j)=1, 1⩽j⩽N-1
Input $f(x_k)$
Compute a_k, b_k, s_{kj}, m_{kj}
%Backward Chebyshev transform
for k=0 to N do
    $g_k = \frac{2}{\tilde{c}_k N} \sum_{j=0}^{N} \frac{1}{\tilde{c}_j} f(x_j) \cos(kj\pi/N)$
endfor
%Evaluate $\bar{f}$ from $f_k = (\sum_{j=0}^{N} g_j T_j(x), \phi_k(x))$
$f_0 = \frac{\pi}{2}(2g_0 + a_0 g_1 + b_0 g_2)$
for k=1 to N-2 do
    $f_k = \frac{\pi}{2}(g_k + a_k g_{k+1} + b_k g_{k+2})$
endfor
Solve $(S+\alpha M)\bar{u} = \bar{f}$
%Evaluate g_k from $\sum_{j=0}^{N-2} \hat{u}_j \phi_j(x) = \sum_{j=0}^{N} g_j T_j(x)$
$g_0 = \hat{u}_0$, $g_1 = \hat{u}_1 + a_0 \hat{u}_0$
for k=2 to N-2 do
    $g_k = \hat{u}_k + a_{k-1}\hat{u}_{k-1} + b_{k-2}\hat{u}_{k-2}$
endfor
$g_{N-1} = a_{N-2}\hat{u}_{N-2} + b_{N-3}\hat{u}_{N-3}$, $g_N = b_{N-2}\hat{u}_{N-2}$
%forward Chebyshev transform
for k=0 to N do
    $\hat{u}_k = \sum_{j=0}^{N} g_j \cos(kj\pi/N)$
end
Output $\hat{u}_0, \hat{u}_1, \ldots, \hat{u}_N$

Exercise 3.3

Problem 1 Repeat the Problem 2 in Section 3.2 with the Chebyshev-Galerkin method.

3.4 Chebyshev-Legendre Galerkin method

The main advantage of using Chebyshev polynomials is that the discrete Chebyshev transforms can be performed in $\mathcal{O}(N \log_2 N)$ operations by using FFT. However, the Chebyshev-Galerkin method leads to non-symmetric and full stiffness matrices. On the other hand, the Legendre-Galerkin method leads to symmetric sparse matrices, but the discrete Legendre transforms are expensive ($\mathcal{O}(N^2)$ operations). In order to take advantages and overcome disadvantages of both the Legendre and Chebyshev polynomials, one may use the so called Chebyshev-Legendre Galerkin method:

$$\alpha \int_I u_N \, v_N \, \mathrm{d}x + \int_I u'_N \, v'_N \, \mathrm{d}x = \int_I I_N^c f v_N \, \mathrm{d}x, \tag{3.4.1}$$

where I_N^c denotes the interpolation operator relative to the Chebyshev-Gauss-Lobatto points. So the only difference with (3.2.3) is that the Chebyshev interpolation operator I_N^c is used here instead of the Legendre interpolation operator in (3.2.3). Thus, as in the Legendre-Galerkin case, (3.4.1) leads to the linear system (3.1.12) with $\bar{u}$, S and M defined in (3.1.10) and (3.2.6) and (3.2.7), but with $\bar{f}$ defined by

$$f_k = \int_I I_N^c f \, \phi_k \mathrm{d}x, \quad \bar{f} = (f_0, f_1, \cdots, f_{N-2})^{\mathrm{T}}. \tag{3.4.2}$$

The solution procedure of (3.4.1) is essentially the same as that of (3.2.3) except that *Chebyshev-Legendre transforms* (between the value of a function at the *CGL* points and the coefficients of its *Legendre* expansion) are needed instead of the *Legendre transforms*. More precisely, given the values of f at the CGL points $\{x_i = \cos(i\pi/N)\}_{0 \leqslant i \leqslant N}$, we determine the values of u_N (solution of (3.1.9)) at the CGL points as follows:

1. (Pre-computation) Compute $\{a_k, b_k\}$ and nonzero elements of S and M;
2. Evaluate the Legendre coefficients of $I_N^c f(x)$ from $\{f(x_i)\}_{i=0}^N$ (backward Chebyshev-Legendre transform);
3. Evaluate $\bar{f}$ from (3.4.2) and solve $\bar{u}$ from (3.1.12);
4. Evaluate $u_N(x_j) = \sum_{i=0}^{N-2} \hat{u}_i \phi_i(x_j)$, $j = 0, 1, \cdots, N$ ("modified" forward Chebyshev-Legendre transform).

The backward and forward Chebyshev-Legendre transforms can be efficiently implemented. Indeed, each Chebyshev-Legendre transform can be split into two steps:

1. The transform between its values at Chebyshev-Gauss-Lobatto points and the coefficients of its Chebyshev expansion. This can be done in $\mathcal{O}(N \log_2 N)$ operations by using FFT.

2. The transform between the coefficients of the Chebyshev expansion and that of the Legendre expansion. Alpert and Rohklin[2] have developed an $\mathcal{O}(N)$-algorithm for this transform, given a prescribed precision.

Therefore, the total computational cost for (3.4.1) is of order $\mathcal{O}(N \log_2 N)$. The algorithm in [2] is based on the fast multipole method (cf. [65]). Hence, it is most attractive for very large N. For moderate N, the algorithm described below appears to be more competitive.

Let us write

$$p(x) = \sum_{i=0}^{N} f_i T_i(x) = \sum_{i=0}^{N} g_i L_i(x),$$
$$\boldsymbol{f} = (f_0, f_1, \cdots, f_N)^{\mathrm{T}}, \quad \boldsymbol{g} = (g_0, g_1, \cdots, g_N)^{\mathrm{T}}.$$

What we need is to transform between $\boldsymbol{f}$ and $\boldsymbol{g}$. The relation between $\boldsymbol{f}$ and $\boldsymbol{g}$ can be easily obtained by computing $(p, T_j)_\omega$ and (p, L_j). In fact, let us denote

$$a_{ij} = \frac{2}{c_i \pi}(T_i, L_j)_\omega, \quad b_{ij} = \left(i + \frac{1}{2}\right)(L_i, T_j),$$

where $c_0 = 2$ and $c_i = 1$ for $i \geqslant 1$, and

$$A = (a_{ij})_{i,j=0}^{N}, \quad B = (b_{ij})_{i,j=0}^{N}.$$

Then we have

$$\boldsymbol{f} = A\boldsymbol{g}, \quad \boldsymbol{g} = B\boldsymbol{f}, \quad AB = BA = I. \tag{3.4.3}$$

By the orthogonality and parity of the Chebyshev and Legendre polynomials, we observe immediately that

$$a_{ij} = b_{ij} = 0, \quad \text{for } i > j \text{ or } i + j \text{ odd}.$$

Hence, both A and B only have about $\frac{1}{4}N^2$ nonzero elements, and the cost of each

transform between $\boldsymbol{f}$ and $\boldsymbol{g}$ is about $\frac{1}{2}N^2$ operations. Consequently, the cost of each Chebyshev-Legendre transform is about $(\frac{5}{2}N\log_2 N + 4N) + \frac{1}{2}N^2$ operations as opposed to $2N^2$ operations for the Legendre transform. In pure operational counts, the cost of the two transforms is about the same at $N = 8$, and the Chebyshev-Legendre transform costs about one third of the Legendre transform at $N = 128$ (see [141] for computational comparisons of the three methods).

The one-dimensional Chebyshev-Legendre transform can be done in about

$$\left(\frac{5}{2}N\log_2 N + 4N\right) + \min\left(\frac{1}{2}N^2, CN\right) \sim \mathcal{O}(N\log_2 N)$$

operations, where C is a large constant in Alpert and Rohklin's algorithm[2]. Since multi-dimensional transforms in the tensor product form are performed through a sequence of one-dimensional transforms, the d-dimensional Chebyshev-Legendre transform can be done in $\mathcal{O}(N^d\log_2 N)$ operations and it has the same speedup as in the 1-D case, when compared with the d-dimensional Legendre transform.

The nonzero elements of A and B can be easily determined by the recurrence relations

$$\begin{aligned} T_{i+1}(x) &= 2xT_i(x) - T_{i-1}(x), \quad i \geqslant 1, \\ L_{i+1}(x) &= \frac{2i+1}{i+1}xL_i(x) - \frac{i}{i+1}L_{i-1}(x), \quad i \geqslant 1. \end{aligned} \tag{3.4.4}$$

Indeed, for $j \geqslant i \geqslant 1$,

$$\begin{aligned} a_{i,j+1} &= (T_i, L_{j+1})_\omega \\ &= \left(T_i, \frac{2j+1}{j+1}xL_j - \frac{j}{j+1}L_{j-1}\right) \\ &= \frac{2j+1}{j+1}(xT_i, L_j)_\omega - \frac{j}{j+1}a_{i,j-1} \\ &= \frac{2j+1}{2j+2}(T_{i+1} + T_{i-1}, L_j)_\omega - \frac{j}{j+1}a_{i,j-1} \\ &= \frac{2j+1}{2j+2}(a_{i+1,j} + a_{i-1,j}) - \frac{j}{j+1}a_{i,j-1}. \end{aligned}$$

Similarly, we have for $j \geqslant i \geqslant 1$,

$$b_{i,j+1} = \frac{2i+2}{2i+1} b_{i+1,j} + \frac{2i}{2i+1} b_{i-1,j} - b_{i,j-1}.$$

Thus, each nonzero element of A and B can be obtained by just a few operations. Furthermore, the Chebyshev-Legendre transform (3.4.3) is extremely easy to implement, while the algorithm in [2] requires considerable programming effort.

Remark 3.4.1 Note that only for equations with constant or polynomial (and rational polynomials in some special cases) coefficients, one can expect the matrices resulting from a Galerkin method to be sparse or have special structure. In the more general cases such as (3.1.5), the Galerkin matrices are usually full, so a direct application of the Galerkin methods is not advisable. However, for many practical situations, the Galerkin system for a suitable constant coefficient problems provides an optimal preconditioner for solving problems with variable coefficients; see Section 3.5 for further details.

Exercise 3.4

Problem 1 Implement the Chebyshev-Legendre transform and find the Legendre expansion coefficients of $T_8(x)$.

Problem 2 Repeat the Problem 2 in Section 3.2 with the Chebyshev-Legendre-Galerkin method.

3.5 Preconditioned iterative method

Preconditioning in frequency space
Condition number estimate —— a special case
Chebyshev case

We now consider the problem with variable coefficients:

$$\mathcal{A}u := -(a(x)u'(x))' + b(x)u = g(x), \tag{3.5.1}$$

subject to the homogeneous boundary conditions:

$$a_- u(-1) + b_- u'(-1) = 0, \quad a_+ u(1) + b_+ u'(1) = 0, \tag{3.5.2}$$

where $a_\pm$ and $b_\pm$ satisfy (2.4.3). We assume that there are three constants c_1, c_2 and c_3 such that

$$0 \leqslant c_1 \leqslant a(x) \leqslant c_2, \qquad 0 \leqslant b(x) \leqslant c_3. \tag{3.5.3}$$

The (weighted) spectral-Galerkin method (3.1.9), including the Legendre- and Chebyshev-Galerkin methods, leads to full stiffness and mass matrices. Hence, it is preferable to solve (3.1.12) using a (preconditioned) iterative method.

Preconditioning in frequency space

Let us take, for example,

$$\bar{a} = \frac{1}{2}(\max_{x\in I} a(x) + \min_{x\in I} a(x)), \quad \bar{b} = \frac{1}{2}(\max_{x\in I} b(x) + \min_{x\in I} b(x))$$

and define $\mathcal{B}u := -\bar{a}u'' + \bar{b}u$. Let S and M be the stiffness and mass matrices associated with $\mathcal{A}$ defined in (3.1.10) and (3.1.11), and $\bar{S}$ and $\bar{M}$ be the stiffness and mass matrices associated with $\mathcal{B}$, i.e., with $a(x) = \bar{a}$ and $b(x) = \bar{b}$ in (3.1.10) and (3.1.11). Then, it can be argued that $S + M$ and $\bar{S} + \bar{M}$ are spectrally equivalent, in the sense that the condition number of $(\bar{S} + \bar{M})^{-1}(S + M)$ is uniformly bounded with respect to the discretization parameter N (see below for a proof of this fact in the Legendre case). Hence, instead of applying a suitable iterative method, e.g, conjugate gradient (CG) in the Legendre case and BiCGSTAB or CGS (cf. [134]; see also Section 1.7) in the Chebyshev case, directly to (3.1.12), we can apply it to the preconditioned system

$$(\bar{S} + \bar{M})^{-1}(S + M)\bar{u} = (\bar{S} + \bar{M})^{-1}\bar{f}. \tag{3.5.4}$$

Thus, to apply a (preconditioned) iterative method for solving (3.5.4), we need to perform the following two processes:

1. Given a vector $\bar{u}$, compute $(S + M)\bar{u}$.
2. Given a vector $\bar{f}$, find $\bar{u}$ by solving $(\bar{S} + \bar{M})\bar{u} = \bar{f}$.

It has been shown in the previous two sections that for both Legendre or Chebyshev Galerkin approximations, the second task can be performed in $\mathcal{O}(N)$ flops.

We now describe how to perform the first task efficiently. Given $\bar{u} = (u_0, u_1, \cdots, u_{N-2})^{\mathrm{T}}$, we set $u_N = \sum_{k=0}^{N-2} \tilde{u}_k\phi_k$. Hence,

$$(S\bar{u})_j = -([I_N(au_N')]', \phi_j)_\omega, \quad 0 \leqslant j \leqslant N-2,$$

and they can be computed as follows (recall that $\phi_k = p_k + a_k p_{k+1} + b_k p_{k+2}$ where $\{p_k\}$ are either the Legendre or Chebyshev polynomials):

1. Use (1.3.5) or (1.3.22d) to determine $\tilde{u}_k^{(1)}$ from

$$u'_N(x) = \sum_{k=0}^{N-2} \tilde{u}_k \phi'_k(x) = \sum_{k=0}^{N} \tilde{u}_k^{(1)} p_k(x);$$

2. (`Forward discrete transform.`) Compute

$$u'_N(x_j) = \sum_{k=0}^{N} \tilde{u}_k^{(1)} p_k(x_j), \quad j = 0, 1, \cdots, N;$$

3. (`Backward discrete transform.`) Determine $\{\tilde{w}_k\}$ from

$$I_N(au'_N)(x_j) = \sum_{k=0}^{N} \tilde{w}_k p_k(x_j), \quad j = 0, 1, \cdots, N;$$

4. Use (1.3.5) or (1.3.22d) to determine $\{\tilde{w}_k^{(1)}\}$ from

$$[I_N(au'_N)]'(x) = \sum_{k=0}^{N} \tilde{w}_k p'_k(x) = \sum_{k=0}^{N} \tilde{w}_k^{(1)} p_k(x);$$

5. For $j = 0, 1, \cdots, N-2$, compute

$$-([I_N(au'_N)]', \phi_j)_\omega = -\sum_{k=0}^{N} \tilde{w}_k^{(1)} (p_k, \phi_j)_\omega.$$

Note that the main cost in the above procedure is the two discrete transforms in Steps 2 and 3. The cost for Steps 1, 4 and 5 are all $\mathcal{O}(N)$ flops.

Similarly, $(M\bar{u})_j = (I_N(bu_N), \phi_j)_\omega$ can be computed as follows:

1. Compute

$$u_N(x_j) = \sum_{k=0}^{N} \tilde{u}_k \phi_k(x_j), \quad j = 0, 1, \cdots, N;$$

2. Determine $\{\tilde{w}_k\}$ from

$$I_N(bu_N)(x_j) = \sum_{k=0}^{N} \tilde{w}_k p_k(x_j), \quad j = 0, 1, \cdots, N;$$

3. Compute

$$(I_N(bu_N), \phi_j)_\omega, \quad j = 0, 1, \cdots, N-2.$$

Hence, if $b(x)$ is not a constant, two additional discrete transforms are needed.

In summary, the total cost for evaluating $(S+M)\bar{u}$ is dominated by four (only two if $b(x)$ is a constant) discrete transforms, and is $\mathcal{O}(N^2)$ (resp. $\mathcal{O}(N\log N)$) flops in the Legendre (resp. the Chebyshev) case. Since the condition number of $(\bar{S}+\bar{M})^{-1}(S+M)$ is uniformly bounded, so is the number of iterations for solving (3.5.4). Hence, the total cost of solving (3.1.12) will be $\mathcal{O}(N^2)$ (and $\mathcal{O}(N\log N)$) flops in the Legendre (and the Chebyshev) case, respectively.

Remark 3.5.1 In the case of Dirichlet boundary conditions, we have $\phi_k(x) = L_k(x) - L_{k+2}(x)$ which, together with (1.3.22a), implies that $\phi_k'(x) = -(2k+3)L_{k+1}(x)$. Therefore, from $u = \sum_{k=0}^{N-2}\tilde{u}_k\phi_k(x)$, we can easily obtain the derivative

$$u' = -\sum_{k=0}^{N-2}(2k+3)\tilde{u}_k L_{k+1}(x)$$

in the frequency space.

Condition number estimate —— a special case

We now show that the condition number of $(\bar{S}+\bar{M})^{-1}(S+M)$ is uniformly bounded in the Legendre case with the Dirichlet boundary condition. The proof for the general case is similar and left as an exercise.

To simplify the proof, we shall replace $(I_N(bu_N),\phi_j)_\omega$ in the Legendre case by the symmetric form $(bu_N,\phi_j)_N$. Due to the exactness of the Legendre-Gauss-Lobatto quadrature, only the term with $j=N$ is slightly changed.

We first remark that $-([I_N(au_N')]',v_N) = (au_N',v_N')_N$. Hence, the matrices S and M (with the above modification) are symmetric. With the notations in (3.1.10), we find

$$\begin{aligned}\langle(S+M)\bar{u},\bar{u}\rangle_{l^2} &= (au_N',v_N')_N + (bu_N,u_N)_N \\ &\leqslant 2\bar{a}(u_N',v_N')_N + 2\bar{b}(u_N,u_N)_N = 2\langle(\bar{S}+\bar{M})\bar{u},\bar{u}\rangle_{l^2}.\end{aligned} \tag{3.5.5}$$

By the Poincaré inequality, there exists $c_4>0$ such that

$$\begin{aligned}\langle(S+M)\bar{u},\bar{u}\rangle_{l^2} &\geqslant c_4\left(\bar{a}(u_N',v_N')_N + 2\bar{b}(u_N,u_N)_N\right) \\ &= c_4\langle(\bar{S}+\bar{M})\bar{u},\bar{u}\rangle_{l^2}.\end{aligned} \tag{3.5.6}$$

Since $(\bar{S}+\bar{M})^{-1}(S+M)$ is symmetric with respect to the inner product $\langle\bar{u},\bar{v}\rangle_{\bar{S}+\bar{M}} := \langle(\bar{S}+\bar{M})\bar{u},\bar{v}\rangle_{l^2}$, we derive immediately that

$$\text{cond}((\bar{S}+\bar{M})^{-1}(S+M)) \leqslant \frac{2}{c_4}. \tag{3.5.7}$$

In other words, $(\bar{S}+\bar{M})^{-1}$ is an optimal preconditioner for B, and the convergence rate of the conjugate gradient method applied to (3.5.4) will be independent of N.

Remark 3.5.2 We make three relevant remarks:

• We recall from Section 3.2 that $(\bar{S}+\bar{M})$ can be efficiently inverted so the main cost is the evaluation of $(S+M)\bar{u}$;

• Due to the Poincaré inequality, (3.5.7) holds if we replace $\bar{S}+\bar{M}$ by $\bar{S}$. In this case, inversion of $\bar{S}$ is negligible since $\bar{S}$ is diagonal;

• Suppose we use the normalized basis function

$$\tilde{\phi}_k := \sqrt{-b_k(4k+6)}^{-1}\phi_k \text{ with } (\tilde{\phi}'_j, \tilde{\phi}'_i) = \delta_{ij}.$$

In this case, no preconditioner is needed since $\text{cond}(S+M)$ is uniformly bounded under this basis. However, if $\bar{b}$ is relatively large with respect to $\bar{a}$, it is more efficient to use $\bar{S}+\bar{M}$ as a preconditioner.

Chebyshev case

In the Chebyshev case, an appropriate preconditioner for the inner product $b_{N,\omega}(u_N, v_N)$ in $X_N \times X_N$ is $(u'_N, \omega^{-1}(v_N\omega)')_\omega$ for which the associated linear system can be solved in $\mathcal{O}(N)$ flops as shown in Section 3.2. Unfortunately, we do not have an estimate similar to (3.5.7) since no coercivity result for $b_{N,\omega}(u_N, v_N)$ is available to the authors' knowledge. However, ample numerical results indicate that the convergence rate of a conjugate gradient type method for non-symmetric systems such as Conjugate Gradient Square (CGS) or BICGSTAB is similar to that in the Legendre case.

The advantage of using the Chebyshev polynomials is of course that the evaluation of $B\bar{u}$ can be accelerated by FFT.

The preconditioning in the frequency space will be less effective if the coefficients $a(x)$ and $b(x)$ have large variations, since the variation of the coefficients is not taken into account in the construction of the preconditioner.

Exercise 3.5

Problem 1 Consider the problem:

$$x^2u - (e^x u_x)_x = f$$

(where f is determined by the exact solution in Problem 2 of Section 3.2) with the following two sets of boundary conditions

$$u(-1) = 0, \qquad u(1) = 1,$$

and

$$u(-1) - u'(-1) = 0, \; u(1) = 1.$$

For each set of the boundary conditions, solve the above equation using the Chebyshev-collocation method (in the strong form). Take $N = 2^i$ with $i = 4, 5, 6, 7, 8, 9$. Plot $\log_{10} \|u - u_N\|_N / \log_{10} N$ for each γ. Explain your results.

Problem 2 Consider the problem:

$$-(a(x)u_x)_x = f, \quad u(-1) - u_x(-1) = 0, \quad u(1) + u_x(1) = 0,$$

with $a(x) = x^2 + 10^k$.

• (a) Construct the matrix B_N of the Legendre-collocation method in the weak form and the matrix A_N of the piecewise linear finite element method.

• (b) For each $k = 0, 1, 2$, list the ratio of the maximum eigenvalue and minimum eigenvalue for $A_N^{-1} B_N$ as well as its condition number with $N = 16, 32, 64, 128$.

• (c) Consider the exact solution $u(x) = \sin 3\pi x + 3\pi x/2$. Use the conjugate gradient iterative method with and without preconditioning to solve the linear system associated with the Legendre-collocation method in the weak form. For each $k = 0, 1, 2$ and $N = 16, 32, 64, 128$, list the iteration numbers needed (for 10-digit accuracy) and the maximum errors at the Gauss-Lobatto points, with and without preconditioning.

3.6 Spectral-Galerkin methods for higher-order equations

Fourth-order equation
Third-order equation

In this section we shall consider spectral-Galerkin methods for two typical higher-order equations with constant coefficients. Problems with variable coefficients can be treated by a preconditioned iterative method, similarly to second-order equations.

Fourth-order equation

Let us consider

$$\begin{aligned} &u^{(4)} - \alpha u'' + \beta u = f, \quad \alpha, \beta > 0,\ x \in I, \\ &u(\pm 1) = u'(\pm 1) = 0, \end{aligned} \tag{3.6.1}$$

where α, β are two non-negative constants. This equation can serve as a model for the clamped rod problem. A semi-implicit time discretization of the important Kuramoto-Sivashinsky equation modeling a flame propagation is also of this form.

The variational formulation for (3.6.1) is: Find $u \in H_0^2(I)$ such that

$$a(u, v) := (u'', v'') + \alpha(u', v') + \beta(u, v) = (f, v), \quad \forall v \in H_0^2(I). \tag{3.6.2}$$

Let us define

$$V_N := P_N \cap H_0^2(I) = \{v \in P_N :\ v(\pm 1) = v_x(\pm 1) = 0\}. \tag{3.6.3}$$

Then, the spectral-Galerkin approximation to (3.6.2) is to find $u_N \in V_N$ such that

$$(u_N'', v_N'') + \alpha(u_N', v_N') + \beta(u_N, v_N) = (I_N f, v_N), \quad \forall v_N \in V_N, \tag{3.6.4}$$

where I_N is the interpolation operator based on the Legendre-Gauss-Lobatto points.

We next give a brief description of the numerical implementation of the above scheme. As we did before, we choose a compact combination of Legendre polynomials $\{\phi_k\}_{k=0}^{N-4}$ as basis function for V_N, i.e.,

$$\phi_k(x) := d_k\Big(L_k(x) - \frac{2(2k+5)}{2k+7} L_{k+2}(x) + \frac{2k+3}{2k+7} L_{k+4}\Big), \tag{3.6.5}$$

with the normalization factor $d_k = 1/\sqrt{2(2k+3)^2(2k+5)}$. One can verify from (1.3.22c) that $\phi_k(\pm 1) = \phi_k'(\pm 1) = 0$. Therefore,

$$V_N = \text{span}\,\{\phi_0,\ \phi_1,\ \cdots,\ \phi_{N-4}\}.$$

By using the recurrence relation and the orthogonality of the Legendre polynomials, we can prove the following results.

Lemma 3.6.1 *We have*

$$a_{kj} = (\phi_j'',\ \phi_k'') = \delta_{kj}, \tag{3.6.6}$$

and the only non-zero elements of $b_{kj} = (\phi_j,\ \phi_k)$, $c_{kj} = (\phi_j', \phi_k')$ *are:*

$$b_{kk} = d_k^2(e_k + h_k^2 e_{k+2} + g_k^2 e_{k+4}),$$

$$
\begin{aligned}
&b_{k,k+2} = b_{k+2,k} = d_k d_{k+2}(h_k e_{k+2} + g_k h_{k+2} e_{k+4}),\\
&b_{k,k+4} = b_{k+4,k} = d_k d_{k+4} g_k e_{k+4},\\
&c_{kk} = -2(2k+3)d_k^2 h_k,\\
&c_{k,k+2} = c_{k+2,k} = -2(2k+3)d_k d_{k+2},
\end{aligned}
\tag{3.6.7}
$$

where

$$
e_k = \frac{2}{2k+1}, \quad g_k = \frac{2k+3}{2k+7}, \quad h_k = -(1+g_k).
$$

Hence, setting

$$
\begin{aligned}
&B = (b_{kj})_{0\leqslant k,j\leqslant N-4}, \quad C = (c_{kj})_{0\leqslant k,j\leqslant N-4},\\
&f_k = (I_N f, \phi_k), \quad \bar{f} = (f_0, f_1, \cdots, f_{N-4})^{\mathrm{T}},\\
&u_N = \sum_{n=0}^{N-4} \tilde{u}_n \phi_n(x), \quad \bar{u} = (\tilde{u}_0, \tilde{u}_1, \cdots, \tilde{u}_{N-4})^{\mathrm{T}},
\end{aligned}
\tag{3.6.8}
$$

the system (3.6.4) is equivalent to the matrix equation

$$
(\alpha B + \beta C + I)\bar{u} = \bar{f}, \tag{3.6.9}
$$

where the non-zero entries of B and C are given in (3.6.7).

It is obvious that B and C are symmetric positive definite matrices. Furthermore, B can be split into two penta-diagonal sub-matrices, and C can be split into two tridiagonal sub-matrices. Hence, the system can be efficiently solved. In particular, there is no equation to solve when $\alpha = \beta = 0$.

In summary: given the values of f at LGL points $\{x_i\}_{0\leqslant i\leqslant N}$, we determine the values of u_N, solution of (3.6.4), at these LGL points as follows:

1. (Pre-computation) Compute LGL points, and nonzero elements of B and C;
2. Evaluate the Legendre coefficients of $I_N f(x)$ from $\{f(x_i)\}_{i=0}^N$ (backward Legendre transform)and evaluate $\bar{f}$ in (3.6.8);
3. Solve $\bar{u}$ from (3.6.9);
4. Determine $\{\hat{u}_j\}_{j=0}^N$ such that $\sum_{j=0}^{N-4} \tilde{u}_j \phi_j(x) = \sum_{j=0}^N \hat{u}_j L_j(x)$;
5. Evaluate $u_N(x_j) = \sum_{i=0}^N \hat{u}_i \phi_i(x_j)$, $j = 0, 1, \cdots, N$ (forward Legendre transform).

Since this process is very similar to that of the Legendre-Galerkin scheme (3.2.3), a pseudo-code can be easily assembled by modifying the pseudo-code LG-PSN-1D.

Third-order equation

Consider the third-order equation

$$\begin{aligned} &\alpha u - \beta u_x - \gamma u_{xx} + u_{xxx} = f, \ \ x \in I = (-1,1), \\ &u(\pm 1) = u_x(1) = 0, \end{aligned} \tag{3.6.10}$$

where α, β, γ are given constants. Without loss of generality, we only consider homogeneous boundary conditions, for non-homogeneous boundary conditions $u(-1) = c_1,\ u(1) = c_2$ and $u_x(1) = c_3$ can be easily handled by considering $v = u - \hat{u}$, where $\hat{u}$ is the unique quadratic polynomial satisfying the non-homogeneous boundary conditions.

Since the leading third-order differential operator is not symmetric, it is natural to use a Petrov-Galerkin method, in which the trial and test functions are chosen differently. In this context, we define the spaces

$$\begin{aligned} V_N &= \{u \in P_N :\ u(\pm 1) = u_x(1) = 0\}, \\ V_N^* &= \{u \in P_N :\ u(\pm 1) = u_x(-1) = 0\}, \end{aligned} \tag{3.6.11}$$

and consider the following Legendre dual-Petrov-Galerkin (LDPG) approximation for (3.6.10): Find $u_N \in V_N$ such that

$$\begin{aligned} &\alpha(u_N, v_N) - \beta(\partial_x u_N, v_N) + \gamma(\partial_x u_N, \partial_x v_N) + (\partial_x u_N, \partial_x^2 v_N) \\ &\qquad = (I_N f, v_N), \quad \forall v_N \in V_N^*. \end{aligned} \tag{3.6.12}$$

The particular choice of V_N^* allows us to integrate by parts freely, without introducing non-zero boundary terms. This is the key for the efficiency of the numerical algorithm.

Let us first take a look at the matrix form of the system (3.6.12). We choose the basis functions:

$$\begin{aligned} \phi_n(x) &= L_n(x) - \frac{2n+3}{2n+5} L_{n+1}(x) - L_{n+2}(x) + \frac{2n+3}{2n+5} L_{n+3}(x); \\ \psi_n(x) &= L_n(x) + \frac{2n+3}{2n+5} L_{n+1}(x) - L_{n+2}(x) - \frac{2n+3}{2n+5} L_{n+3}(x), \end{aligned} \tag{3.6.13}$$

which satisfy $\phi_n(\pm 1) = \psi_n(\pm 1) = \phi_n'(1) = \psi_n'(-1) = 0$. For $N \geqslant 3$, we have

$$\begin{aligned} V_N &= \text{span}\{\phi_0, \phi_1, \cdots, \phi_{N-3}\}; \\ V_N^* &= \text{span}\{\psi_0, \psi_1, \cdots, \psi_{N-3}\}. \end{aligned} \tag{3.6.14}$$

Hence, by setting

$$u_N = \sum_{k=0}^{N-3} \tilde{u}_k \phi_k, \quad \bar{u} = (\tilde{u}_0, \tilde{u}_1, \cdots, \tilde{u}_{N-3})^{\mathrm{T}},$$

$$\begin{aligned} &\tilde{f}_k = (I_N f, \psi_k), \quad \bar{f} = (\tilde{f}_0, \tilde{f}_1, \cdots, \tilde{f}_{N-3})^{\mathrm{T}}, \\ &m_{ij} = (\phi_j, \psi_i), \quad p_{ij} = -(\phi_j', \psi_i), \quad q_{ij} = (\phi_j', \psi_i'), \quad s_{ij} = (\phi_j', \psi_i''), \end{aligned} \tag{3.6.15}$$

the linear system (3.6.12) becomes

$$(\alpha M + \beta P + \gamma Q + S)\bar{u} = \bar{f}, \tag{3.6.16}$$

where M, P, Q and S are $(N-3) \times (N-3)$ matrices with entries m_{ij}, p_{ij}, q_{ij} and s_{ij}, respectively.

Owing to the orthogonality of the Legendre polynomials, we have $m_{ij} = 0$ for $|i-j| > 3$. Therefore, M is a seven-diagonal matrix. We note that the homogeneous "dual" boundary conditions satisfied by ϕ_j and ψ_i allow us to integrate by parts freely, without introducing additional boundary terms. In other words, we have

$$s_{ij} = (\phi_j', \psi_i'') = (\phi_j''', \psi_i) = -(\phi_j, \psi_i''').$$

Because of the compact form of ϕ_j and ψ_i, we have $s_{ij} = 0$ for $i \neq j$. So S is a diagonal matrix. Similarly, we see that P is a penta-diagonal matrix and Q is a tridiagonal matrix. It is an easy matter to show that

$$s_{ii} = 2(2i+3)^2. \tag{3.6.17}$$

The non-zero elements of M, P, Q can be easily determined from the properties of Legendre polynomials. Hence, the linear system (3.6.16) can be readily formed and inverted.

In summary: given the values of f at LGL points $\{x_i\}_{0 \leqslant i \leqslant N}$, we determine the values of u_N, solution of (3.6.12) at these LGL points as follows:

1. (Pre-computation) Compute the LGL points, and the nonzero elements of M, P, Q and S;
2. Evaluate the Legendre coefficients of $I_N f(x)$ from $\{f(x_i)\}_{i=0}^N$ (backward Legendre transform) and evaluate $\bar{f}$ in (3.6.15);
3. Solve $\bar{u}$ from (3.6.16);
4. Determine $\{\hat{u}_j\}_{j=0}^N$ such that $\sum_{j=0}^{N-3} \tilde{u}_j \phi_j(x) = \sum_{j=0}^N \hat{u}_j L_j(x)$;
5. Evaluate $u_N(x_j) = \sum_{i=0}^N \hat{u}_i \phi_i(x_j)$, $j = 0, 1, \cdots, N$ (forward Legendre transform).

Once again, this process is very similar to that of the Legendre-Galerkin scheme (3.2.3), so a pseudo-code can be easily assembled by modifying the pseudo-code LG-PSN-1D.

One can verify that the basis functions (3.6.13) are in fact generalized Jacobi polynomials:

$$\phi_n(x) = \frac{2n+3}{2(n+1)} j_{n+3}^{-2,-1}(x); \tag{3.6.18a}$$

$$\psi_n(x) = \frac{2n+3}{2(n+1)} j_{n+3}^{-1,-2}(x). \tag{3.6.18b}$$

Exercise 3.6

Problem 1 Solve the equation (3.6.1) using the Legendre-Galerkin method (3.6.4). Take $\alpha = \beta = 1$ and the exact solution $u(x) = \sin^2(4\pi x)$.

Problem 2 Design a Chebyshev-Galerkin method for (3.6.1).

Problem 3 Determine the non-zero entries of M, P and Q in (3.6.15).

Problem 4 Design a dual-Petrov Legendre Galerkin method for the first-order equation

$$\alpha u + u_x = f, \ x \in (-1, 1); \quad u(-1) = 0. \tag{3.6.19}$$

3.7 Error estimates

Legendre-Galerkin method with Dirichlet boundary conditions
Chebyshev-collocation method with Dirichlet boundary conditions
Legendre-Galerkin method for a fourth-order equation
Dual-Petrov Legendre-Galerkin method for a third-order equation

In this section, we present error analysis for four typical cases of the spectral-Galerkin methods presented in previous sections. The error analysis for other cases may be derived in a similar manner. We refer to the books[11, 29, 146] for more details.

The error analysis below relies essentially on the optimal error estimates for various projection/interpolation operators presented in Section 1.8.

Legendre-Galerkin method with Dirichlet boundary conditions

We consider the Legendre-Galerkin approximation of (3.2.1) with homogeneous Dirichlet boundary conditions.

Theorem 3.7.1 *Let u and u_N be respectively the solutions of* (3.2.1) *and* (3.2.3) *with homogeneous Dirichlet boundary conditions. Then, for $u \in H^m_{\omega^{-1,-1},*}(I)$ with $m \geqslant 1$ and $f \in H^k_{\omega^{0,0},*}(I)$ with $k \geqslant 1$, we have*

$$\|\partial_x(u-u_N)\| \lesssim N^{1-m}\|\partial_x^m u\|_{\omega^{m-1,m-1}} + N^{-k}\|\partial_x^k f\|_{\omega^{k,k}}; \tag{3.7.1}$$

$$\|u-u_N\| \lesssim N^{-m}\|\partial_x^m u\|_{\omega^{m-1,m-1}} + N^{-k}\|\partial_x^k f\|_{\omega^{k,k}}. \tag{3.7.2}$$

Proof In this case, we have

$$X_N = \{u \in P_N : u(\pm 1) = 0\}.$$

We observe from the definition of $\pi_{N,\omega^{-1,-1}}$ in (1.8.19) that for $u \in H^1_0(I) \cap L^2_{\omega^{-1,-1}}(I)$ we have

$$\begin{aligned} &(\partial_x(u-\pi_{N,\omega^{-1,-1}}u), \partial_x v_N) = -(u-\pi_{N,\omega^{-1,-1}}u, \partial_x^2 v_N) \\ =& -(u-\pi_{N,\omega^{-1,-1}}u, \omega^{1,1}\partial_x^2 v_N)_{\omega^{-1,-1}} = 0, \qquad \forall v_N \in X_N. \end{aligned} \tag{3.7.3}$$

In other words, $\pi_{N,\omega^{-1,-1}}$ is also the orthogonal projector from $H^1_0(I)$ to X_N associated with the bilinear form $(\partial_x\cdot, \partial_x\cdot)$. Hence, we derive from (3.2.1) and (3.2.3) that

$$\begin{aligned} &\alpha(\pi_{N,\omega^{-1,-1}}u - u_N, v_N) + (\partial_x(\pi_{N,\omega^{-1,-1}}u - u_N), \partial_x v_N) \\ =&(f - I_N f, v_N) + \alpha(\pi_{N,\omega^{-1,-1}}u - u, v_N), \quad \forall v_N \in X_N. \end{aligned} \tag{3.7.4}$$

Taking $v_N = \pi_{N,\omega^{-1,-1}}u - u_N$ in the above, we find

$$\begin{aligned} &\alpha\|\pi_{N,\omega^{-1,-1}}u - u_N\|^2 + \|\partial_x(\pi_{N,\omega^{-1,-1}}u - u_N)\|^2 \\ =&(f - I_N f, \pi_{N,\omega^{-1,-1}}u - u_N) + \alpha(\pi_{N,\omega^{-1,-1}}u - u, \pi_{N,\omega^{-1,-1}}u - u_N). \end{aligned} \tag{3.7.5}$$

Using the Cauchy-Schwarz inequality and Poincaré inequality, we get

$$\begin{aligned}&\alpha\|\pi_{N,\omega^{-1,-1}}u-u_N\|^2+\frac{1}{2}\|\partial_x(\pi_{N,\omega^{-1,-1}}u-u_N)\|^2\\ \lesssim&\|f-I_Nf\|^2+\|\pi_{N,\omega^{-1,-1}}u-u\|^2.\end{aligned}\tag{3.7.6}$$

Then (3.7.1) and (3.7.2) in the case of $\alpha > 0$, can be derived from the triangle inequality and Theorems 1.8.2 and 1.8.4. If $\alpha = 0$, we need to use a standard duality argument which we now describe.

First of all, we derive from (3.2.1) and (3.2.3) with $\alpha = 0$ that

$$((u-u_N)_x,(v_N)_x)=(f-I_Nf,v_N),\quad \forall v_N\in X_N.\tag{3.7.7}$$

Now, consider the dual problem

$$-w_{xx}=u-u_N,\qquad w(\pm1)=0.\tag{3.7.8}$$

Taking the inner product of the above with $u-u_N$, thanks to (3.7.7), (3.7.3) and Theorem 1.8.2, we obtain

$$\begin{aligned}\|u-u_N\|^2&=(w_x,(u-u_N)_x)=((w-\pi_{N,\omega^{-1,-1}}w)_x,(u-u_N)_x)\\ &\quad+(f-I_Nf,\pi_{N,\omega^{-1,-1}}w)\\ &\leqslant\|(w-\pi_{N,\omega^{-1,-1}}w)_x\|\|(u-u_N)_x\|+\|f-I_Nf\|\|\pi_{N,\omega^{-1,-1}}w\|\\ &\lesssim N^{-1}\|w_{xx}\|\|(u-u_N)_x\|+\|f-I_Nf\|(\|w-pi_{N,\omega^{-1,-1}}w\|+\|w\|)\\ &=\|u-u_N\|(N^{-1}\|(u-u_N)_x\|+\|f-I_Nf\|),\end{aligned}$$

which implies that $\|u-u_N\|\lesssim N^{-1}\|(u-u_N)_x\|+\|f-I_Nf\|$. Then (3.7.2) is a direct consequence of the above and (3.7.1). □

Chebyshev-collocation method with Dirichlet boundary conditions

To simplify the notation, we shall use ω to denote the Chebyshev weight $(1-x^2)^{-\frac{1}{2}}$ in this part of the presentation. An essential element in the analysis of the Chebyshev method for the second-order equations with Dirichlet boundary conditions is to show that the bilinear form

$$a_\omega(u,v):=(u_x,\omega^{-1}(v\omega)_x)_\omega=\int_{-1}^{1}u_x(v\omega)_x\mathrm{d}x\tag{3.7.9}$$

is continuous and coercive in $H^1_{0,\omega}(I)\times H^1_{0,\omega}(I)$. To this end, we first need to estab-

lish the following inequality of Hardy type:

Lemma 3.7.1 *For any $u \in H^1_{0,\omega}(I)$ with $\omega = (1-x^2)^{-\frac{1}{2}}$, we have*

$$\int_{-1}^{1} u^2(1+x^2)\omega^5 \mathrm{d}x \leqslant \int_{-1}^{1} u_x^2 \omega \mathrm{d}x. \tag{3.7.10}$$

Proof For any $u \in H^1_{0,\omega}(I)$, we find by integration by parts that

$$\begin{aligned} 2\int_{-1}^{1} u_x u x\omega^3 \mathrm{d}x &= \int_{-1}^{1} (u^2)_x x\omega^3 \mathrm{d}x \\ = -\int_{-1}^{1} u^2 (x\omega^3)_x \mathrm{d}x &= -\int_{-1}^{1} u^2(1+2x^2)\omega^5 \mathrm{d}x. \end{aligned} \tag{3.7.11}$$

Hence,

$$\begin{aligned} 0 &\leqslant \int_{-1}^{1} (u_x + ux\omega^2)^2 \omega \mathrm{d}x \\ &= \int_{-1}^{1} u_x^2 \omega \mathrm{d}x + \int_{-1}^{1} u^2 x^2 \omega^5 \mathrm{d}x + 2\int_{-1}^{1} u_x u x\omega^3 \mathrm{d}x \\ &= \int_{-1}^{1} u_x^2 \omega \mathrm{d}x - \int_{-1}^{1} u^2(1+x^2)\omega^5 \mathrm{d}x. \end{aligned}$$

This completes the proof of this lemma. □

Lemma 3.7.2 *We have*

$$\begin{aligned} &a_\omega(u,v) \leqslant 2\|u_x\|_\omega \|v_x\|_\omega, \quad && \forall u,\, v \in H^1_{0,\omega}(I), \\ &a_\omega(u,u) \geqslant \frac{1}{4}\|u_x\|_\omega^2, && \forall u \in H^1_{0,\omega}(I). \end{aligned}$$

Proof Using the Cauchy-Schwarz inequality, the identity $\omega_x = x\omega^3$ and (3.7.10), we have, for all $u, v \in H^1_{0,\omega}(I)$,

$$\begin{aligned} a_\omega(u,v) &= \int_{-1}^{1} u_x(v_x + vx\omega^2)\omega \mathrm{d}x \\ &\leqslant \|u_x\|_\omega \|v_x\|_\omega + \|u_x\|_\omega \left(\int_{-1}^{1} v^2 x^2 \omega^5 \mathrm{d}x\right)^{\frac{1}{2}} \leqslant 2\|u_x\|_\omega \|v_x\|_\omega. \end{aligned}$$

On the other hand, due to (3.7.11) and (3.7.10), we find

$$\begin{aligned} a_\omega(u,u) &= \int_{-1}^{1} u_x^2 \omega \mathrm{d}x + \int_{-1}^{1} u u_x x \omega^3 \mathrm{d}x \\ &= \|u_x\|_\omega^2 - \frac{1}{2}\int_{-1}^{1} u^2(1+2x^2)\omega^5 \mathrm{d}x \\ &\geqslant \|u_x\|_\omega^2 - \frac{3}{4}\int_{-1}^{1} u^2(1+x^2)\omega^5 \mathrm{d}x \geqslant \frac{1}{4}\|u_x\|_\omega^2, \qquad \forall u \in H^1_{0,\omega}(I). \end{aligned} \tag{3.7.12}$$

The proof of this lemma is complete. □

Thanks to the above lemma, we can define a new orthogonal projector from $H^1_{0,\omega}$ to P_N^0 based on the bilinear form $a_\omega(\cdot,\cdot)$ (note the difference with $\pi^{1,0}_{N,\omega}: \ H^1_{0,\omega} \to X_N = \{v \in P_N : v(\pm 1) = 0\}$ defined in Section 1.8).

Definition 3.7.1 $\tilde{\pi}^{1,0}_{N,\omega}: \ H^1_{0,\omega} \to \ X_N$ is defined by

$$a_\omega(u - \tilde{\pi}^{1,0}_{N,\omega}u, v_N) = \int_{-1}^{1} (u - \tilde{\pi}^{1,0}_{N,\omega}u)'(v_N\omega)' \mathrm{d}x = 0, \ \text{ for } \ v_N \in X_N. \tag{3.7.13}$$

Similar to Theorem 1.8.3, we have the following results:

Lemma 3.7.3 *For any $u \in H^1_{0,\omega}(I) \cap H^m_{\omega^{-\frac{3}{2},-\frac{3}{2}},*}$, we have*

$$\|u - \tilde{\pi}^{1,0}_{N,\omega}u\|_{\nu,\omega} \lesssim N^{\nu-m}\|\partial_x^m u\|_{\omega^{m-\frac{3}{2},m-\frac{3}{2}}}, \qquad \nu = 0, 1. \tag{3.7.14}$$

Proof Using the definition (3.7.13) and Lemma 3.7.2, we find

$$\begin{aligned} \|u - \tilde{\pi}^{1,0}_{N,\omega}u\|^2_{1,\omega} &\lesssim |u - \tilde{\pi}^{1,0}_{N,\omega}u|^2_{1,\omega} \lesssim a_\omega(u - \tilde{\pi}^{1,0}_{N,\omega}u, u - \tilde{\pi}^{1,0}_{N,\omega}u) \\ &= a_\omega(u - \tilde{\pi}^{1,0}_{N,\omega}u, u - \pi^{1,0}_{N,\omega}u) \\ &\leqslant 2|u - \tilde{\pi}^{1,0}_{N,\omega}u|_{1,\omega}|u - \pi^{1,0}_{N,\omega}u|_{1,\omega}. \end{aligned}$$

We then derive (3.7.14) with $\nu = 1$ from above and Theorem 1.8.3. To prove the result with $\nu = 0$, we use again the standard duality argument by considering the dual problem

$$-\phi_{xx} = u - \tilde{\pi}^{1,0}_{N,\omega}u, \qquad \phi(\pm 1) = 0. \tag{3.7.15}$$

Its variational formulation is: find $\phi \in H^1_{0,\omega}(I)$ such that

$$(\phi', (\psi\omega)') = (u - \tilde{\pi}^{1,0}_{N,\omega}u, \psi\omega), \qquad \forall \psi \in H^1_{0,\omega}(I). \tag{3.7.16}$$

According to Lemma 3.7.2, there exists a unique solution $\phi \in H^1_{0,\omega}(I)$ for the above problem, and furthermore, we derive from (3.7.15) that $\phi \in H^2_\omega(I)$ and

$$\|\phi\|_{2,\omega} \lesssim \|u - \tilde{\pi}^{1,0}_{N,\omega}u\|_\omega. \tag{3.7.17}$$

Now we take $\psi = u - \tilde{\pi}^{1,0}_{N,\omega}u$ in (3.7.16). Hence, by Lemma 3.7.2, (3.7.17), (3.7.13) and (3.7.14) with $\nu = 1$,

$$\begin{aligned}
&(u - \tilde{\pi}^{1,0}_{N,\omega}u, u - \tilde{\pi}^{1,0}_{N,\omega}u)_\omega = \int_{-1}^{1} \phi_x((u - \tilde{\pi}^{1,0}_{N,\omega}u)\omega)_x \mathrm{d}x \\
&= \int_{-1}^{1} (\phi - \tilde{\pi}^{1,0}_{N,\omega}\phi)_x((u - \tilde{\pi}^{1,0}_{N,\omega}u)\omega)_x \mathrm{d}x \leqslant 2|\phi - \tilde{\pi}^{1,0}_{N,\omega}\phi|_{1,\omega}|u - \tilde{\pi}^{1,0}_{N,\omega}u|_{1,\omega} \\
&\lesssim N^{-1}\|\phi\|_{2,\omega}|u - \tilde{\pi}^{1,0}_{N,\omega}u|_{1,\omega} \lesssim N^{-1}\|u - \tilde{\pi}^{1,0}_{N,\omega}u\|_\omega|u - \tilde{\pi}^{1,0}_{N,\omega}u|_{1,\omega}.
\end{aligned}$$

The above and (3.7.14) with $\nu = 1$ conclude the proof. □

We are now in the position to establish an error estimate of the Chebyshev-collocation method for (3.2.1) which reads:

$$\begin{aligned}
&\alpha u_N(x_j) - u''_N(x_j) = f(x_j), \qquad j = 1, \cdots, N-1, \\
&u_N(x_0) = u_N(x_N) = 0,
\end{aligned} \tag{3.7.18}$$

where $x_j = \cos(j\pi/N)$. To this end, we need to rewrite (3.7.18) in a suitable variational formulation by using the discrete inner product (1.2.23) associated with the Chebyshev-Gauss-Lobatto quadrature (1.2.22). One verifies easily that for $v_N \in X_N$, we have $\omega^{-1}(v_N\omega)' \in P_{N-1}$. Therefore, thanks to (1.2.22), we find that

$$(u'_N, \omega^{-1}(v_N\omega)')_{N,\omega} = (u'_N, \omega^{-1}(v_N\omega)')_\omega = -(u''_N, v_N)_\omega = -(u''_N, v_N)_{N,\omega}. \tag{3.7.19}$$

Let $\{h_k(x)\}_{k=0}^N$ be the Lagrange interpolation polynomials associated with $\{x_k\}_{k=0}^N$ and take the discrete inner product of (3.7.18) with $h_k(x)$ for $k = 1, \cdots, N-1$. Thanks to (3.7.19) and the fact that

$$X_N = \text{span}\{h_1(x), h_2(x), \cdots, h_{N-1}(x)\},$$

we find that the solution u_N of (3.7.18) verifies:

$$\alpha(u_N, v_N)_{N,\omega} + a_\omega(u_N, v_N) = (I_{N,\omega}f, v_N)_{N,\omega}, \qquad \forall v_N \in X_N. \tag{3.7.20}$$

Theorem 3.7.2 *Let u and u_N be respectively the solutions of* (3.2.1) *and* (3.7.20). *Then, for $u \in H^m_{\omega^{-\frac{3}{2},-\frac{3}{2}},*}(I)$ with $m \geqslant 1$ and $f \in H^k_{\omega^{-\frac{1}{2},-\frac{1}{2}},*}(I)$ with $k \geqslant 1$, we have*

$$\|u-u_N\|_{1,\omega} \lesssim N^{1-m}\|\partial_x^m u\|_{\omega^{m-3/2,m-3/2}} + N^{-k}\|\partial_x^k f\|_{\omega^{k-\frac{1}{2},k-\frac{1}{2}}}, \qquad m,k \geqslant 1, \tag{3.7.21}$$

$$\|u-u_N\|_{\omega} \lesssim N^{-m}\|\partial_x^m u\|_{\omega^{m-1/2,m-1/2}} + N^{-k}\|\partial_x^k f\|_{\omega^{k-\frac{1}{2},k-\frac{1}{2}}}, \qquad m,k \geqslant 1. \tag{3.7.22}$$

Proof Using (3.2.1) and (3.7.13) we obtain

$$\alpha(u,v_N)_\omega + a_\omega(\tilde{\pi}^{1,0}_{N,\omega}u, v_N) = (f,v_N)_\omega, \ \ \forall v_N \in X_N.$$

Hence, for all $v_N \in X_N$,

$$\begin{aligned}
&\alpha(\tilde{\pi}^{1,0}_{N,\omega}u - u_N, v_N)_{N,\omega} + a_\omega(\tilde{\pi}^{1,0}_{N,\omega}u - u_N, v_N)\\
=&(f,v_N)_\omega - (I_{N,\omega}f, v)_{N,\omega} + \alpha(\tilde{\pi}^{1,0}_{N,\omega}u, v_N)_{N,\omega} - \alpha(u,v_N)_\omega\\
=&(f-\pi_{N,\omega}f, v_N)_\omega - (I_{N,\omega}f - \pi_{N,\omega}f, v)_{N,\omega}\\
&+\alpha(\tilde{\pi}^{1,0}_{N,\omega}u - \pi_{N,\omega}u, v_N)_{N,\omega} - \alpha(u-\pi_{N,\omega}u, v_N)_\omega,
\end{aligned} \tag{3.7.23}$$

where the operator $\pi_{N,\omega} = \pi_{N,\omega^{-\frac{1}{2},-\frac{1}{2}}}$ is defined in (1.8.6). Hence, taking $v_N = \tilde{\pi}^{1,0}_{N,\omega}u - u_N$ in the above formula, we find, by the Cauchy-Schwarz inequality and Lemma 3.7.2,

$$\begin{aligned}
&\alpha\|\tilde{\pi}^{1,0}_{N,\omega}u - u_N\|^2_\omega + |\tilde{\pi}^{1,0}_{N,\omega}u - u_N|^2_{1,\omega}\\
\lesssim&\|f-\pi_{N,\omega}f\|^2_\omega + \|f - I_{N,\omega}f\|^2_\omega + \alpha\|u-\tilde{\pi}^{1,0}_{N,\omega}u\|^2_\omega + \alpha\|u-\pi_{N,\omega}u\|^2_\omega.
\end{aligned} \tag{3.7.24}$$

It follows from Theorems 1.8.1, 1.8.4 and (3.7.14) that

$$\begin{aligned}
\|u-u_N\|_{1,\omega} &\leqslant \|u-\tilde{\pi}^{1,0}_{N,\omega}u\|_{1,\omega} + \|\tilde{\pi}^{1,0}_{N,\omega}u - u_N\|_{1,\omega}\\
&\lesssim N^{1-m}\|\partial_x^m u\|_{\omega^{m-3/2,m-3/2}} + N^{-k}\|\partial_x^k f\|_{\omega^{k-\frac{1}{2},k-\frac{1}{2}}}.
\end{aligned}$$

For $\alpha > 0$, we can also derive (3.7.22) directly from (3.7.24), while for $\alpha = 0$, a duality argument is needed. The details are left as an exercise. □

Legendre-Galerkin method for a fourth-order equation

We now consider the error estimate for (3.6.4). Let V_N be defined in (3.6.3). Before we proceed with the error estimates, it is important to make the following observation:

$$\begin{aligned}&(\partial_x^2(\pi_{N,\omega^{-2,-2}}u-u),\partial_x^2 v_N)=(\pi_{N,\omega^{-2,-2}}u-u,\partial_x^4 v_N)\\ =&(\pi_{N,\omega^{-2,-2}}u-u,\omega^{2,2}\partial_x^4 v_N)_{\omega^{-2,-2}}=0,\quad \forall v_N\in V_N.\end{aligned}\tag{3.7.25}$$

Hence, $\pi_{N,\omega^{-2,-2}}$ is also the orthogonal projector from $H_0^2(I)$ to V_N.

It is also important to note that the basis functions given in (3.6.5) are in fact the generalized Jacobi polynomials with index $(-2,-2)$. More precisely, we find from (1.4.12) that ϕ_k defined in (3.6.5) is proportional to the generalized Jacobi polynomial $J_{k+4}^{-2,-2}(x)$. This relation allows us to perform the error analysis for the proposed scheme (3.6.4) by using Theorem 1.8.2.

Theorem 3.7.3 *Let u and u_N be respectively the solution of* (3.6.1) *and* (3.6.4)*. If $u\in H_0^2(I)\cap H^m_{\omega^{-2,-2},*}(I)$ with $m\geqslant 2$ and $f\in H^k_{\omega^{0,0},*}(I)$ with $k\geqslant 1$, then*

$$\|\partial_x^l(u-u_N)\|\lesssim N^{l-m}\|\partial_x^m u\|_{\omega^{m-2,m-2}}+N^{-k}\|\partial_x^k f\|_{\omega^{k,k}},\quad 0\leqslant l\leqslant 2.\tag{3.7.26}$$

Proof Using (3.6.2) and (3.6.4) leads to the error equation

$$\begin{aligned}a(u-u_N,v_N)=&a(\pi_{N,\omega^{-2,-2}}u-u_N,v_N)-a(\pi_{N,\omega^{-2,-2}}u-u,v_N)\\ =&(f-I_N f,v_N),\qquad \forall v_N\in V_N.\end{aligned}$$

Let us denote $\hat{e}_N=\pi_{N,\omega^{-2,-2}}u-u_N$, $\tilde{e}_N=u-\pi_{N,\omega^{-2,-2}}u$ and $e_N=u-u_N=\tilde{e}_N+\hat{e}_N$.

Taking $v_N=\hat{e}_N$ in the above equality, we obtain from (3.7.25) that

$$\|\hat{e}_N''\|^2+\alpha\|\hat{e}_N'\|^2+\beta\|\hat{e}_N\|^2=\alpha(\tilde{e}_N',\hat{e}_N')+\beta(\tilde{e}_N,\hat{e}_N)+(f-I_N f,\hat{e}_N).$$

Since $(\tilde{e}_N',\hat{e}_N')=-(\tilde{e}_N,\hat{e}_N'')$, it follows from the Cauchy-Schwarz inequality and the Poincaré inequality that

$$\|\hat{e}_N''\|^2+\alpha\|\hat{e}_N'\|^2+\beta\|\hat{e}_N\|^2\lesssim\|\tilde{e}_N\|^2+\|f-I_N f\|^2.$$

We obtain immediately the result for $l=2$ from the triangular inequality and Theorems 1.8.2 and 1.8.4. The results for $l=0,\,1$ can also be directly derived if $\alpha,\,\beta>0$. For $\alpha=0$ or $\beta=0$, a duality argument is needed for the cases with $l=0,\,1$. The

details are left as an exercise. □

Dual-Petrov Legendre-Galerkin method for a third-order equation

The last problem we consider in this section is the error between the solutions of (3.6.10) and (3.6.12). However, although the dual-Petrov-Galerkin formulation (3.6.12) is most suitable for implementation, it is more convenient in terms of error analysis to reformulate (3.6.12) into a suitable equivalent form.

Let V_N and $V_N^\star$ be defined in (3.6.11). Notice that for any $u_N \in V_N$ we have $\omega^{-1,1}u_N \in V_N^*$. Thus, the dual-Petrov-Galerkin formulation (3.6.12) is equivalent to the following weighted spectral-Galerkin approximation: Find $u_N \in V_N$ such that

$$\begin{aligned}&\alpha(u_N, v_N)_{\omega^{-1,1}} - \beta(\partial_x u_N, v_N)_{\omega^{-1,1}} + \gamma(\partial_x u_N, \omega^{1,-1}\partial_x(v_N\omega^{-1,1}))_{\omega^{-1,1}}\\ &+ (\partial_x u_N, \omega^{1,-1}\partial_x^2(v_N\omega^{-1,1}))_{\omega^{-1,1}} = (I_N f, v_N)_{\omega^{-1,1}},\ \ \forall v_N \in V_N.\end{aligned} \tag{3.7.27}$$

We shall show first that the problem (3.7.27) is well-posed. To this end, let us first prove the following generalized Poincaré inequalities:

Lemma 3.7.4 *For any $u \in V_N$, we have*

$$\begin{aligned}\int_I u^2(1-x)^{-4}\mathrm{d}x &\leqslant \frac{4}{9}\int_I u_x^2(1-x)^{-2}\mathrm{d}x,\\ \int_I u^2(1-x)^{-3}\mathrm{d}x &\leqslant \int_I u_x^2(1-x)^{-1}\mathrm{d}x.\end{aligned} \tag{3.7.28}$$

Proof Let $u \in V_N$ and $h \leqslant 2$. Then, for any constant q, we have

$$\begin{aligned}0 &\leqslant \int_I \Big(\frac{u}{1-x} + qu_x\Big)^2 \frac{1}{(1-x)^h}\mathrm{d}x\\ &= \int_I \Big(\frac{u^2}{(1-x)^{2+h}} + q\frac{(u^2)_x}{(1-x)^{1+h}} + q^2\frac{u_x^2}{(1-x)^h}\Big)\mathrm{d}x\\ &= (1-(1+h)q)\int_I \frac{u^2}{(1-x)^{2+h}}\mathrm{d}x + q^2\int_I \frac{u_x^2}{(1-x)^h}\mathrm{d}x.\end{aligned}$$

We obtain the first inequality in (3.7.28) by taking $h = 2$ and $q = \frac{2}{3}$, and the second inequality with $h = 1$ and $q = 1$. □

Remark 3.7.1 Note that with the change of variable $x \to -x$ in the above lemma, we can establish corresponding inequalities for $u \in V_N^*$.

The leading third-order differential operator is coercive in the following sense:

Lemma 3.7.5 *For any $u \in V_N$, we have*

$$\frac{1}{3}\|u_x\|^2_{\omega^{-2,0}} \leqslant (u_x, (u\omega^{-1,1})_{xx}) \leqslant 3\|u_x\|^2_{\omega^{-2,0}}. \tag{3.7.29}$$

Proof For any $u \in V_N$, we have $u\omega^{-1,1} \in V_N^*$. Since the homogeneous boundary conditions are built into the spaces V_N and V_N^*, all the boundary terms from the integration by parts of the third-order term vanish. Therefore, using the identity $\partial_x^k \omega^{-1,1}(x) = 2\,k!(1-x)^{-(k+1)}$ and Lemma 3.7.4, we find

$$\begin{aligned}
&(u_x, (u\omega^{-1,1})_{xx}) = (u_x, u_{xx}\omega^{-1,1} + 2u_x\omega_x^{-1,1} + u\omega_{xx}^{-1,1})\\
=&\frac{1}{2}\int_I \big((u_x^2)_x\omega^{-1,1} + (u^2)_x\omega_{xx}^{-1,1} + 4u_x^2\omega_x^{-1,1}\big)\,\mathrm{d}x = \int_I \Big(\frac{3}{2}u_x^2\omega_x^{-1,1} - \frac{1}{2}u^2\omega_{xxx}^{-1,1}\Big)\mathrm{d}x\\
=&3\int_I \frac{u_x^2}{(1-x)^2}\mathrm{d}x - 6\int_I \frac{u^2}{(1-x)^4}\mathrm{d}x \geqslant \frac{1}{3}\int_I \frac{u_x^2}{(1-x)^2}\mathrm{d}x.
\end{aligned}$$

The desired results follow immediately from the above. □

Before we proceed with the error estimates, we make the following simple but important observation:

$$\begin{aligned}
&(\partial_x(u - \pi_{N,\omega^{-2,-1}}u), \partial_x^2 v_N)\\
=&-(u - \pi_{N,\omega^{-2,-1}}u, \omega^{2,1}\partial_x^3 v_N)_{\omega^{-2,-1}} = 0, \quad \forall u \in V,\ v_N \in V_N^*,
\end{aligned} \tag{3.7.30}$$

where $\pi_{N,\omega^{-2,-1}}$ is defined in (1.8.19).

Theorem 3.7.4 *Let u and u_N be respectively the solution of* (3.6.10) *and* (3.6.12). *Let $\alpha > 0$, $\beta \geqslant 0$ and $-\frac{1}{3} < \gamma < \frac{1}{6}$. Then, for $u \in H^m_{\omega^{-2,-1},*}(I)$ with $m \geqslant 2$ and $f \in H^k_{\omega^{0,0},*}(I)$ with $k \geqslant 1$, we have*

$$\begin{aligned}
&\alpha\|e_N\|_{\omega^{-1,1}} + N^{-1}\|(e_N)_x\|_{\omega^{-1,0}}\\
\lesssim&(1+|\gamma|N)N^{-m}\|\partial_x^m u\|_{\omega^{m-2,m-1}} + N^{-k}\|\partial_x^k f\|_{\omega^{k,k}}.
\end{aligned} \tag{3.7.31}$$

Proof Let us define $\hat{e}_N = \pi_{N,\omega^{-2,-1}}u - u_N$ and $e_N = u - u_N = (u - \pi_{N,\omega^{-2,-1}}u) + \hat{e}_N$. We derive from (3.6.10), (3.7.27) and (3.7.30) that

$$\begin{aligned}
&\alpha(e_N, v_N)_{\omega^{-1,1}} - \beta(\partial_x e_N, v_N)_{\omega^{-1,1}} + \gamma(\partial_x e_N, \omega^{1,-1}\partial_x(v_N\omega^{-1,1}))_{\omega^{-1,1}}\\
&+(\partial_x\hat{e}_N, \omega^{1,-1}\partial_x^2(v_N\omega^{-1,1}))_{\omega^{-1,1}}\\
=&\,(f - I_N f, v_N)_{\omega^{-1,1}}, \qquad \forall v_N \in V_N.
\end{aligned} \tag{3.7.32}$$

Taking $v_N = \hat{e}_N$ in the above, using Lemma 3.7.5 and the identities

$$\begin{aligned} -(v_x, v)_{\omega^{-1,1}} &= -\frac{1}{2}\int_I (v^2)_x \omega^{-1,1} \mathrm{d}x = \|v\|^2_{\omega^{-2,0}}, \qquad \forall v \in V_N, \\ (v_x, (v\omega^{-1,1})_x) &= (v_x, v_x\omega^{-1,1} + 2v\omega^{-2,0}) = \|v_x\|^2_{\omega^{-1,1}} - 2\|v\|^2_{\omega^{-3,0}}, \qquad \forall v \in V_N, \end{aligned} \tag{3.7.33}$$

we obtain

$$\begin{aligned} &\alpha\|\hat{e}_N\|^2_{\omega^{-1,1}} + \beta\|\hat{e}_N\|^2_{\omega^{-2,0}} + \gamma\|(\hat{e}_N)_x\|^2_{\omega^{-1,1}} - 2\gamma\|\hat{e}_N\|^2_{\omega^{-3,0}} + \frac{1}{3}\|(\hat{e}_N)_x\|^2_{\omega^{-2,0}} \\ \leqslant& -\alpha(u - \pi_{N,\omega^{-2,-1}}u, \hat{e}_N)_{\omega^{-1,1}} + \beta(\partial_x(u - \pi_{N,\omega^{-2,-1}}u), \hat{e}_N)_{\omega^{-1,1}} \\ &- \gamma(\partial_x(u - \pi_{N,\omega^{-2,-1}}u), \partial_x(\hat{e}_N\omega^{-1,1})) + (f - I_N f, \hat{e}_N)_{\omega^{-1,1}}. \end{aligned}$$

The right-hand side can be bounded by using Lemma 3.7.4, the Cauchy-Schwarz inequality and the fact that $\omega^{-1,2} \leqslant 2\omega^{-1,1} \leqslant 2\omega^{-2,0}$ in I:

$$\begin{aligned} (u - \pi_{N,\omega^{-2,-1}}u, \hat{e}_N)_{\omega^{-1,1}} &\leqslant \|\hat{e}_N\|_{\omega^{-1,1}}\|u - \pi_{N,\omega^{-2,-1}}u\|_{\omega^{-1,1}} \\ &\lesssim \|\partial_x\hat{e}_N\|_{\omega^{-2,0}}\|u - \pi_{N,\omega^{-2,-1}}u\|_{\omega^{-2,-1}}, \end{aligned}$$

$$\begin{aligned} ((u - \pi_{N,\omega^{-2,-1}}u)_x, \hat{e}_N)_{\omega^{-1,1}} &= -(u - \pi_{N,\omega^{-2,-1}}u, \partial_x\hat{e}_N\omega^{-1,1} + 2\hat{e}_N\omega^{-2,0}) \\ &\lesssim \|u - \pi_{N,\omega^{-2,-1}}u\|_{\omega^{-2,-1}}\|\partial_x\hat{e}_N\|_{\omega^{-2,0}}, \end{aligned}$$

$$\begin{aligned} ((u - \pi_{N,\omega^{-2,-1}}u)_x, (\hat{e}_N\omega^{-1,1})_x) &= ((u - \pi_{N,\omega^{-2,-1}}u)_x, (\hat{e}_N)_x\omega^{-1,1} + 2\hat{e}_N\omega^{-2,0}) \\ &\leqslant \|(u - \pi_{N,\omega^{-2,-1}}u)_x\|_{\omega^{-1,0}}\|\partial_x\hat{e}_N\|_{\omega^{-2,0}}, \\ (f - I_N f, \hat{e}_N)_{\omega^{-1,1}} &\leqslant \|f - I_N f\|\|\hat{e}_N\|_{\omega^{-2,2}} \\ &\leqslant \|f - I_N f\|\|\partial_x\hat{e}_N\|_{\omega^{-2,0}}. \end{aligned}$$

For $0 \leqslant \gamma < \frac{1}{6}$, we choose δ sufficiently small such that $\frac{1}{3} - 2\gamma - \delta > 0$. Combining the above inequalities, using the inequality

$$ab \leqslant \epsilon a^2 + \frac{1}{4\epsilon}b^2, \quad \forall \epsilon > 0, \tag{3.7.34}$$

Theorem 1.8.4, and dropping some unnecessary terms, we get

$$\begin{aligned} &\alpha\|\hat{e}_N\|^2_{\omega^{-1,1}} + \left(\frac{1}{3} - 2\gamma - \delta\right)\|(\hat{e}_N)_x\|^2_{\omega^{-2,0}} \\ \lesssim& \|u - \pi_{N,\omega^{-2,-1}}u\|^2_{\omega^{-2,-1}} + \gamma\|(u - \pi_{N,\omega^{-2,-1}}u)_x\|^2_{\omega^{-1,0}} + \|f - I_N f\|^2 \\ \lesssim& (1 + \gamma N^2)N^{-2m}\|\partial_x^m u\|_{\omega^{m-2,m-1}} + N^{-k}\|\partial_x^k f\|_{\omega^{k,k}}. \end{aligned}$$

The last inequality follows from Theorem 1.8.2. For $-\frac{1}{3} < \gamma < 0$, we choose δ sufficiently small such that $\frac{1}{3} + \gamma - \delta > 0$, and we derive similarly

$$\begin{aligned}&\alpha\|\hat{e}_N\|^2_{\omega^{-1,1}} + \left(\frac{1}{3} + \gamma - \delta\right)\|(\hat{e}_N)_x\|^2_{\omega^{-2,0}}\\ \lesssim&(1 + |\gamma|N^2)N^{-2m}\|\partial_x^m u\|_{\omega^{m-2,m-1}} + N^{-k}\|\partial_x^k f\|_{\omega^{k,k}}.\end{aligned}$$

On the other hand, we derive from the triangle inequality, Theorem 1.8.2, and $\|u\|_{\omega^{-1,0}} \leqslant 2\|u\|_{\omega^{-2,0}}$ that

$$\begin{aligned}&\|(e_N)_x\|_{\omega^{-1,0}} \leqslant \|(\hat{e}_N)_x\|_{\omega^{-1,0}} + \|(u - \pi_{N,\omega^{-2,-1}}u)_x\|_{\omega^{-1,0}}\\ \lesssim&\|(\hat{e}_N)_x\|_{\omega^{-2,0}} + N^{1-m}\|\partial_x^m u\|_{\omega^{m-2,m-1}}.\end{aligned}$$

Then, the desired results follow from the above and triangular inequalities. □

Exercise 3.7

Problem 1 Prove (3.7.22) in the case of $\alpha = 0$, using a duality argument as in the proof of Lemma 3.7.3.

Problem 2 Prove (3.7.26) with $l = 0, 1$ in the cases of $\alpha = \beta = 0$, using a duality argument.

Problem 3 Continue with Problem 4 in Section 3.6. Perform the corresponding error analysis.

Problem 4 Continue with Problem 5 in Section 3.6. Perform the corresponding error analysis.

Chapter 4

Spectral Methods in Unbounded Domains

Contents

4.1 Hermite spectral methods 144

4.2 Laguerre spectral methods 158

4.3 Spectral methods using rational functions 170

4.4 Error estimates in unbounded domains 177

In the previous chapters, we discussed various spectral methods for problems in bounded intervals or with periodic boundary conditions. In this chapter, we shall present spectral methods for unbounded domains.

As before, spectral methods in unbounded domains will also be based on orthogonal polynomials or orthogonal functions in the underlying domain. Hence, instead of Jacobi polynomials or Fourier series, we will use Hermite polynomials/functions, Laguerre polynomials/functions and some rational orthogonal functions. In Section 4.1, we begin with some properties of the Hermite polynomials/functions. Then we discussed the Hermite-collocation and Hermite-Galerkin methods. In Section 4.2, Laguerre spectral methods in a semi-infinite interval will be investigated. The Laguerre-collocation and Galerkin methods will be applied to solve differential equations with general boundary conditions. In Section 4.3, rational spectral methods in a semi-infinite interval will be considered, which are particularly suitable for problems

whose solutions do not decay exponentially to zero as $|x| \to \infty$. Finally, in Section 4.4, we will discuss some basic technqiues for obtaining error bounds for spectral methods in unbounded domains.

For unbounded domains, proper scaling factors are necessary. Here the scaling parameter α is defined by the change of variable $x = \alpha\tilde{x}$. We will discuss how to make optimal choices of such a scaling factor.

Some of the references on spectral methods in unbounded domains include [11], [90], [74], [144] for Laguerre methods, [54], [137], [72], [45] for Hermite methods, and [19], [**?**], [105], [76] for rational functions.

4.1 Hermite spectral methods

Hermite polynomials
Hermite functions
Interpolation, discrete transform and derivatives
Hermite-collocation and Hermite-Galerkin methods
Scaling factors
Numerical experiments

For problems posed on $\mathbb{R} := (-\infty, +\infty)$, one immediately thinks of the classical Hermite polynomials/functions. We begin with some properties of the Hermite polynomials/functions.

Hermite polynomials

The Hermite polynomials, denoted by $H_n(x), n \geqslant 0, x \in \mathbb{R}$, are defined by the following three-term recurrence relation:

$$\begin{aligned} &H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x), \quad n \geqslant 1, \\ &H_0(x) = 1, \quad H_1(x) = 2x. \end{aligned} \tag{4.1.1}$$

They are orthogonal with respect to the weight function $\omega(x) = e^{-x^2}$:

$$\int_{\mathbb{R}} H_m(x)H_n(x)e^{-x^2}\mathrm{d}x = \gamma_n\delta_{mn}, \quad \gamma_n = \sqrt{\pi}2^n n!. \tag{4.1.2}$$

We list below some basic properties of the Hermite polynomials, which can be found, for instance, in [155].

- Hermite polynomials are eigenfunctions of the Sturm-Liouville problem:

$$e^{x^2}(e^{-x^2}H_n'(x))' + 2nH_n(x) = 0. \tag{4.1.3}$$

- Derivative relations:

$$H_n'(x) = 2nH_{n-1}(x), \quad n \geqslant 1, \tag{4.1.4a}$$
$$H_n'(x) = 2xH_n(x) - H_{n+1}(x), \quad n \geqslant 0. \tag{4.1.4b}$$

- It follows from (4.1.2) and (4.1.4a) that

$$\int_{\mathbb{R}} H_n'(x)H_m'(x)\omega(x)\mathrm{d}x = 4n^2\gamma_{n-1}\delta_{mn}. \tag{4.1.5}$$

- The leading coefficient of $H_n(x)$ is $k_n = 2^n$.
- Odd-even symmetries:

$$H_{2n}(-x) = H_{2n}(x), \quad H_{2n+1}(-x) = -H_{2n+1}(x), \tag{4.1.6a}$$
$$H_{2n}(0) = (-1)^n\frac{(2n)!}{n!}, \quad H_{2n+1}(0) = 0. \tag{4.1.6b}$$

We now introduce the Hermite-Gauss quadrature.

Theorem 4.1.1 (Hermite-Gauss) *Let $\{x_j\}_{j=0}^N$ be the zeros of $H_{N+1}(x)$, and set*

$$\omega_j = \frac{\sqrt{\pi}2^N N!}{(N+1)H_N^2(x_j)}, \quad 0 \leqslant j \leqslant N. \tag{4.1.7}$$

Then,

$$\int_{-\infty}^{\infty} p(x)e^{-x^2}\mathrm{d}x = \sum_{j=0}^{N} p(x_j)\omega_j, \quad \forall p \in P_{2N+1}. \tag{4.1.8}$$

According to Theorem 1.2.1, $\{x_j\}_{j=0}^N$ are the eigenvalues of an $(N+1)\times(N+1)$ symmetric tridiagonal matrix (1.2.5) with

$$\alpha_j = 0, \quad 0 \leqslant j \leqslant N; \qquad \beta_j = \frac{j}{2}, \quad 1 \leqslant j \leqslant N. \tag{4.1.9}$$

We note from (4.1.6) and (4.1.7) that the nodes and weights are symmetric,

$$x_j = -x_{N-j}, \quad \omega_j = \omega_{N-j}, \qquad 0 \leqslant j \leqslant N, \tag{4.1.10}$$

and $x_{N/2} = 0$, if N even. Moreover, it can be shown that (cf. [102])

$$\omega_j \sim \frac{1}{N} e^{-x_j^2} \Big(1 - \frac{|x_j|}{\sqrt{2(N+1)}}\Big), \quad 0 \leqslant j \leqslant N. \tag{4.1.11}$$

Hence, the ω_j are exponentially small for large x_j. Thus, the Hermite Gauss quadrature (4.1.8) is not suitable in most practical computations since (i) it is for a weighted inner product with an exponentially decaying weight, and (ii) it is difficult to compute $p(x_j)$ and ω_j accurately when j and N are large. Therefore, one should use the so-called Hermite functions.

Hermite functions

The normalized Hermite function of degree n is defined by

$$\widehat{H}_n(x) = \frac{1}{\sqrt{2^n n!}} e^{-x^2/2} H_n(x), \quad n \geqslant 0, \ \ x \in \mathbb{R}. \tag{4.1.12}$$

Clearly, $\{\widehat{H}_n\}$ is an orthogonal system in $L^2(\mathbb{R})$, i.e.,

$$\int_{\mathbb{R}} \widehat{H}_n(x) \widehat{H}_m(x) \mathrm{d}x = \sqrt{\pi} \delta_{mn}. \tag{4.1.13}$$

The three-term recurrence relation (4.1.1) implies

$$\begin{aligned} &\widehat{H}_{n+1}(x) = x\sqrt{\frac{2}{n+1}} \widehat{H}_n(x) - \sqrt{\frac{n}{n+1}} \widehat{H}_{n-1}(x), \ \ n \geqslant 1, \\ &\widehat{H}_0(x) = e^{-x^2/2}, \quad \widehat{H}_1(x) = \sqrt{2} x e^{-x^2/2}. \end{aligned} \tag{4.1.14}$$

Property (4.1.4a) and the above formula lead to

$$\begin{aligned} \partial_x \widehat{H}_n(x) &= \sqrt{2n} \widehat{H}_{n-1}(x) - x \widehat{H}_n(x) \\ &= \sqrt{\frac{n}{2}} \widehat{H}_{n-1}(x) - \sqrt{\frac{n+1}{2}} \widehat{H}_{n+1}(x), \end{aligned} \tag{4.1.15}$$

and this implies

$$\int_{\mathbb{R}} \partial_x \widehat{H}_n \partial_x \widehat{H}_m \mathrm{d}x = \begin{cases} -\dfrac{\sqrt{n(n-1)\pi}}{2}, & m = n-2, \\ \sqrt{\pi}\left(n + \dfrac{1}{2}\right), & m = n, \\ -\dfrac{\sqrt{(n+1)(n+2)\pi}}{2}, & m = n+2, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1.16}$$

In contrast to the Hermite polynomials, the normalized Hermite functions are well-behaved, since

$$\max_{|x|\geqslant\epsilon} |\widehat{H}_n(x)| \sim n^{-1/12}, \quad \forall \epsilon > 0. \tag{4.1.17}$$

This behavior is demonstrated in Figure 4.1.

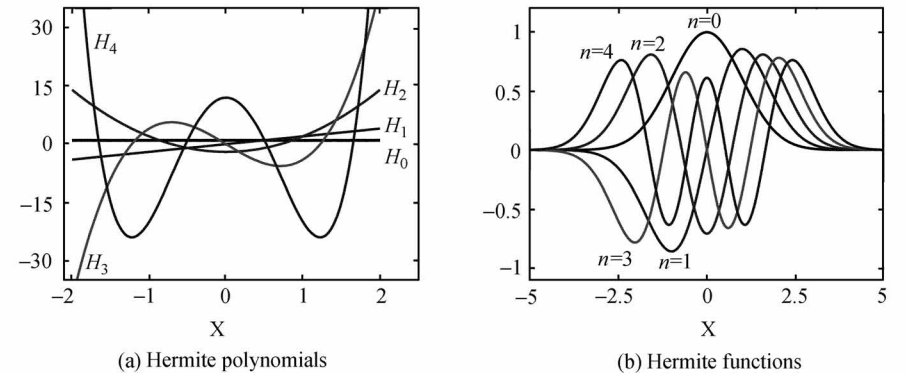

Figure 4.1

(a) Hermite polynomials $H_n(x)$ with $n = 0, \cdots, 4$; (b) Hermite functions $\widehat{H}_n(x)$ with $n = 0, \cdots, 4$.

It is straightforward to derive from Theorem 4.1.1 the Gauss quadrature associated with the Hermite functions.

Let $\{x_j\}_{j=0}^N$ be the Hermite-Gauss nodes and define the weights

$$\hat{\omega}_j = \frac{\sqrt{\pi}}{(N+1)\widehat{H}_N^2(x_j)}, \quad 0 \leqslant j \leqslant N. \tag{4.1.18}$$

Then we have

$$\int_{\mathbb{R}} p(x)\mathrm{d}x = \sum_{j=0}^{N} p(x_j)\hat{\omega}_j, \quad \forall p \in \{u : \ u = e^{-x^2} v, \ v \in P_{2N+1}\}. \tag{4.1.19}$$

Interpolation, discrete transform and derivatives

We define the function space

$$\hat{P}_N := \{u \ : \ u = e^{-x^2/2} v, \ \ \forall v \in P_N\}, \tag{4.1.20}$$

and denote by $\hat{I}_N$ the interpolation operator in $\hat{P}_N$ based on the Hermite-Gauss points

$\{x_j\}_{j=0}^N$, i.e., for all $u \in C(\mathbb{R})$,

$$\hat{I}_N u(x_j) = u(x_j), \quad 0 \leqslant j \leqslant N. \tag{4.1.21}$$

For any $u \in \hat{P}_N$, we write

$$u(x) = \sum_{n=0}^{N} \hat{u}_n \widehat{H}_n(x), \qquad u'(x) = \sum_{n=0}^{N} \hat{u}_n^{(1)} \widehat{H}_n(x). \tag{4.1.22}$$

By Hermite-Gauss quadrature, the interpolation coefficients $\{\hat{u}_n\}$ are determined by

$$\hat{u}_n = \frac{\sqrt{\pi}}{N+1} \sum_{j=0}^{N} \frac{\widehat{H}_n(x_j)}{\widehat{H}_N^2(x_j)} u(x_j), \ \ 0 \leqslant n \leqslant N. \tag{4.1.23}$$

The recurrence relation (4.1.15) is used to find $\{\hat{u}_n^{(1)}\}_{n=0}^N$ from the coefficients $\{\hat{u}_n\}_{n=0}^N$, as follows:

$$\hat{u}_N^{(1)} = \sqrt{\frac{N}{2}} \hat{u}_{N-1}; \quad \hat{u}_k^{(1)} = \sqrt{\frac{k+1}{2}} \hat{u}_{k+1} - \sqrt{\frac{k}{2}} \hat{u}_{k-1}, \quad 0 \leqslant k \leqslant N-1, \tag{4.1.24}$$

with the understanding that $\hat{u}_{-1}^{(0)} = 0$.

We now turn to the differentiation matrix. For $u \in \hat{P}_N$, we can write

$$u(x) = \sum_{j=0}^{N} u(x_j) \hat{h}_j(x) \in \hat{P}_N, \tag{4.1.25}$$

where $\{\hat{h}_j\}_{j=0}^N$ are Lagrange interpolation functions, defined by

$$\hat{h}_j(x) = \frac{e^{-x^2/2}}{e^{-x_j^2/2}} \frac{H_{N+1}(x)}{(x - x_j) H'_{N+1}(x_j)}. \tag{4.1.26}$$

Hence, the entries of the first-order differentiation matrix $\widehat{D}$ can be computed by the formula:

$$\hat{d}_{kj} = \hat{h}'_j(x_k) = \begin{cases} \dfrac{\widehat{H}_N(x_k)}{\widehat{H}_N(x_j)} \dfrac{1}{x_k - x_j} & \text{if } \ k \neq j, \\ 0 & \text{if } \ k = j. \end{cases} \tag{4.1.27}$$

Hermite-collocation and Hermite-Galerkin method

To illustrate how to solve differential equations in $\mathbb{R}$ using Hermite functions, we consider the following eigenvalue problem (cf. [13]):

$$-u''(x) + x^4 u(x) = \lambda u(x), \qquad x \in \mathbb{R}, \quad u(x) \to 0 \text{ as } |x| \to \infty. \tag{4.1.28}$$

`Hermite-collocation method` Let $\{x_j\}_{j=0}^N$ be the set of Hermite-Gauss points. The Hermite-collocation method for (4.1.28) is to find $u_N \in \hat{P}_N$ and λ s.t.

$$-u_N''(x_j) + x_j^4 u_N(x_j) = \lambda u_N(x_j), \quad j = 0, 1, \cdots, N. \tag{4.1.29}$$

Set $\bar{u} = (u_N(x_0), u_N(x_1), \cdots, u_N(x_N))^{\mathrm{T}}$. Then (4.1.29) can be written in the form

$$(\widehat{D}^2 + \mathrm{diag}(x_0^4, x_1^4, \cdots, x_N^4))\bar{u} = \lambda \bar{u}, \tag{4.1.30}$$

where $\widehat{D}$ is the $(N+1)\times(N+1)$ matrix defined in (4.1.27) and $\mathrm{diag}(x_0^4, x_1^4, \cdots, x_N^4)$ is the diagonal matrix with diagonal entries being $x_0^4, x_1^4, \cdots, x_N^4$.

`Hermite-Galerkin method` The Hermite-Galerkin method for (4.1.28) is to find $u_N \in \hat{P}_N$ and λ such that

$$(u_N', \widehat{H}_j') + (x^4 u_N, \widehat{H}_j) = \lambda (u_N, \widehat{H}_j), \quad j = 0, 1, \cdots, N. \tag{4.1.31}$$

Hence, by setting

$$u_N = \sum_{k=0}^{N} \hat{u}_k \widehat{H}_k(x), \quad \bar{u} = (\hat{u}_0, \hat{u}_1, \cdots, \hat{u}_N)^{\mathrm{T}},$$

$$s_{ik} = (\widehat{H}_k', \ \widehat{H}_i'), \qquad S = (s_{ik})_{0 \leqslant i,k \leqslant N},$$

$$m_{ik} = (x^4 \widehat{H}_k, \ \widehat{H}_i), \quad M = (m_{ik})_{0 \leqslant i,k \leqslant N},$$

the system (4.1.31) can be reduced to the matrix form

$$\Big(S + M\Big)\bar{u} = \sqrt{\pi}\lambda \bar{u}. \tag{4.1.32}$$

We note that S is a symmetric positive matrix with three non-zero diagonals given in (4.1.16). We derive from (4.1.14) that $m_{ik} = m_{ki} = 0$ if $|i-k| > 4$. The non-zero entries can be determined from (4.1.14) and (4.1.13). They can also be

"approximated" by the Gauss quadrature, namely,

$$m_{ik} = (x^4 \widehat{H}_k,\ \widehat{H}_i) \approx \sum_{j=0}^{N} x_j^4 \widehat{H}_i(x_j)\widehat{H}_k(x_j)\hat{\omega}_j.$$

Hence, (4.1.32) can be efficiently solved.

Scaling factors

Suppose that the function u has a finite support $[-M, M]$, i.e. $u(x) \sim 0$ for $|x| > M$. In order to compute $\{\hat{u}_n\}_{n=0}^N$ by (4.1.23) we need to use information from the interval $[-M, M]$ only, since outside this region the function is almost zero and will not give much contribution to $\hat{u}_n$. This simple motivation suggests us to scale the grid through the transform $y = \frac{1}{\alpha_N} x$ so that we have the collocation points in y satisfying

$$|y_j| = \left| \frac{x_j}{\alpha_N} \right| \leqslant M, \qquad \text{for} \quad \text{all} \quad 0 \leqslant j \leqslant N, \tag{4.1.33}$$

where $\{x_j\}$ are the roots of $H_{N+1}(x)$. It is clear that the above condition is satisfied by choosing the scaling factor

$$\alpha_N = \max_{0 \leqslant j \leqslant N} \{x_j\}/M = x_N/M. \tag{4.1.34}$$

Let us now examine the effect of scaling through a specific example. Many practical problems require the approximation of the distribution function of the form $\exp(-pv^2)$ with moderate and large values of p. Due to the parity of the Hermite polynomials, we can write

$$\exp(-px^2) = \sum_{n=0}^{\infty} c_{2n} \widehat{H}_{2n}(x), \tag{4.1.35}$$

where the coefficients c_{2n} can be computed explicitly,

$$c_{2n} = \frac{(-1)^n}{\sqrt{2^{2n}(2n)!(p+1/2)}} \left(\frac{p-1/2}{p+1/2} \right)^n \frac{(2n)!}{n!}. \tag{4.1.36}$$

We would like to determine how many terms are needed in (4.1.35) for the truncation error to be sufficiently small. It can be shown that, asymptotically, we have

$$c_{2n} \sim \frac{1}{\sqrt{n\pi p}} \left(\frac{p-1/2}{p+1/2} \right)^n. \tag{4.1.37}$$

Since

$$\left(1-\frac{1}{x}\right)^x \leqslant \lim_{a\to\infty}\left(1-\frac{1}{a}\right)^a = \frac{1}{e}, \quad \text{for} \quad \text{all} \quad x \geqslant 1,$$

then, only when $n \geqslant N \approx Cp$ with a positive constant C, we have

$$c_{2n} \sim \frac{1}{\sqrt{n\pi p}} e^{-C}. \tag{4.1.38}$$

Hence, given an accuracy threshold ϵ, we should choose $C = -\log\epsilon$, i.e., we need $N = \mathcal{O}(-p\log\epsilon)$ terms.

Now let us consider the expansion in scaled Hermite functions,

$$\exp(-px^2) = \sum_{n=0}^{\infty} d_{2n}\widehat{H}_{2n}(\alpha_N x), \tag{4.1.39}$$

with $\alpha_N = x_N/M$ as given in (4.1.34) and the asymptotic behavior

$$d_{2n} \sim \frac{\alpha_N}{\sqrt{n\pi p}}\left(\frac{p/\alpha_N^2 - 1/2}{p/\alpha_N^2 + 1/2}\right)^n. \tag{4.1.40}$$

Since $x_N \sim \sqrt{2N}$ (see e.g. [1]) we obtain

$$\alpha_N \sim \sqrt{2N}/M, \tag{4.1.41}$$

and

$$d_{2n} \sim \sqrt{\frac{2N}{n\pi p M^2}}\left(\frac{M^2 p/(2N) - 1/2}{M^2 p/(2N) + 1/2}\right)^n. \tag{4.1.42}$$

For p large we can set $M^2 = 2$ for the sake of simplicity. Hence, for $n \geqslant C(p/N+1)$, we have

$$\begin{aligned} d_{2n} &\sim \sqrt{\frac{N}{n\pi p}}\left(1-\frac{2}{p/N+1}\right)^n \\ &\lesssim \frac{N}{p}\left(1-\frac{2}{p/N+1}\right)^{C(p/N+1)} \leqslant \frac{N}{p}e^{-C}. \end{aligned} \tag{4.1.43}$$

Thus, for an accuracy threshold ϵ, we should choose $C = -\log\epsilon$, so the requirement $N = C(p/N+1)$ is satisfied when $N = \mathcal{O}(\sqrt{-p\log\epsilon})$. Hence, much fewer terms are needed when a proper scaling is used.

`Time-dependent scaling` In [45], Hermite spectral methods were inves-

tigated for linear second-order partial differential equations and the viscous Burgers' equation in unbounded domains. When the solution domain is unbounded, the diffusion operator no longer has a compact resolvent, which makes the Hermite spectral methods unstable. To overcome this difficulty, a time-dependent scaling factor was employed in the Hermite expansions, which yields a positive bilinear form. As a consequence, stability and spectral convergence were established for this approach [45]. The method in [45] plays a similar stability role to the similarity transformation technique proposed by Funaro and Kavian[54]. However, since coordinate transformations are not required, this approach is more efficient and is easier to implement. In fact, with the time-dependent scaling the resulting discretization system is of the same form as that associated with the classical (straightforward but unstable) Hermite spectral method.

Below we present a Petrov-Galerkin Hermite spectral method with a time-dependent weight function for the following simple parabolic problem:

$$\begin{aligned} &\partial_t u - \nu \partial_x^2 u = f(x,t), \quad x \in \mathbb{R}, \quad t > 0, \\ &\lim_{|x|\to\infty} u(x,t) = 0, \quad t > 0; \quad u(x,0) = u_0(x), \quad x \in \mathbb{R}, \end{aligned} \tag{4.1.44}$$

where the constant $\nu > 0$. Let $P_N(\mathbb{R})$ be the space of polynomials of degree at most N, $\omega_\alpha = \exp(-(\alpha x)^2)$, $\alpha = \alpha(t) > 0$ is a function of t, and

$$V_N = \{v_N(x) = \omega_\alpha \phi_N(x) \mid \phi_N(x) \in P_N(\mathbb{R})\}. \tag{4.1.45}$$

The semi-discrete Hermite function method for (4.1.44) is to find $u_N(t) \in V_N$ such that for any $\varphi_N \in P_N(\mathbb{R})$,

$$\begin{aligned} (\partial_t u_N(t), \varphi_N) + \nu(\partial_x u_N(t), \partial_x \varphi_N) = (f(t), \varphi_N), \quad t > 0, \\ (u_N(0), \varphi_N) = (u_0, \varphi_N), \end{aligned} \tag{4.1.46}$$

where $(\cdot,\cdot)$ is the inner product in the space $L^2(\mathbb{R})$. It was proposed in [45] that

$$\alpha(t) = \frac{1}{2\sqrt{\nu\delta_0(\delta t + 1)}}, \tag{4.1.47}$$

where δ_0 and δ are some positive parameters. To simplify the computation, let

$$u_N(x,t) = \frac{\omega_\alpha}{\sqrt{\pi}} \sum_{l=0}^{N} \hat{u}_l(t) H_l(\alpha x), \qquad \varphi_N(x,t) = (2^m m!)^{-1} \alpha(t) H_m(\alpha x)$$

$$(0 \leqslant m \leqslant N). \tag{4.1.48}$$

In other words, we expand the unknown solution in terms of scaled Hermite functions, and the scaling is now dependent on time.

Theorem 4.1 (cf. [45]) *Let u and u_N be the solutions of* (4.1.44) *and* (4.1.46)*, respectively. Assume that $U \in C(0,T;H^{\sigma}_{\omega_\alpha^{-1}}(\mathbb{R}))$ $(\sigma \geqslant 1)$, and the weight function α defined by* (4.1.47) *with $2\delta_0\delta > 1$. Then*

$$\|u_N(t) - u(t)\|_{\omega_\alpha^{-1}} \leqslant CN^{-\sigma/2}, \quad \forall 0 < t < T, \tag{4.1.49}$$

where C is a constant independent of N.

Numerical experiments

Example 4.1.1 *The first example is the problem* (4.1.28):

$$-u''(x) + x^4u(x) = \lambda u(x), \qquad x \in \mathbb{R}.$$

By the WKB method[83], the solution of the above equation has the asymptotic behavior

$$u(x) \sim \exp(-|x|^3/3). \tag{4.1.50}$$

It is obvious from (4.1.50) that $u \sim 0$ if $|x| \geqslant M \approx 5$. In order to obtain accurate solutions of (4.1.28) efficiently, we need to choose the scaling factor $\alpha = x_0/M$, where $x_0 = \max_{0\leqslant j\leqslant N}\{x_j\}$, with x_j the roots of $H_{N+1}(x)$. Since the solutions of (4.1.28) are even functions, only $N/2$ expansion terms are required in the actual calculations. With $N = 60$ we predict the scaling factor $\alpha \approx 10.16/5.0 \approx 2.0$. Birkhoff and Fix[13] used Galerkin method with 30 Hermite functions (i.e. $N = 60$) to solve (4.1.28). They found that the standard Hermite functions (i.e. without scaling) gave the first 18 eigenvalues to only three decimal places, whereas using a scaling factor $\alpha = 2.154$ gave the same eigenvalues to 10 decimal places. That is, an increase of 10^{-7} in accuracy is obtained. They obtained the optimum scaling factor through trial and error (the procedure requires a considerable amount of computer time), but the theory in the last subsection provides an accurate scaling factor in a very simple way.

Example 4.1.2 *Consider the heat equation*

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\Big(\nu\frac{\partial u}{\partial x}\Big), \quad x \in \mathbb{R}, \tag{4.1.51}$$

where ν is the viscosity coefficient, with the initial distribution

$$u(x,0) = \frac{1}{\sqrt{\nu\pi}} \exp\left(-x^2/\nu\right). \tag{4.1.52}$$

If the viscosity coefficient is a constant, then the exact solution of (4.1.51) and (4.1.52) is

$$u(x,t) = \frac{1}{\sqrt{\pi\nu(4t+1)}} \exp\left(-\frac{x^2}{\nu(4t+1)}\right). \tag{4.1.53}$$

Problem (4.1.51) and (4.1.52) has been chosen since it has an analytic solution and this allows us to compare our numerical results with the exact solution (4.1.53). It can be seen from the previous subsections that the Hermite spectral methods will work well for moderate values of ν, but about $\mathcal{O}(1/\nu)$ expansion terms are needed when ν is small. However, if we apply a proper scaling then much fewer terms are required. To illustrate this we shall consider the case when $\nu = 0.01$. It is pointed out that the numerical procedure can be applied to more complicated initial distributions and to variable viscosity coefficients.

Let $\bar{u}(t) = (u(x_0/\alpha_N, t), \cdots, u(x_N/\alpha_N, t))^{\mathrm{T}}$. Then a semi-discretization of (4.1.51) is

$$\frac{d\bar{u}}{dt} = \nu\widehat{D}^2\bar{u}, \tag{4.1.54}$$

where $\widehat{D}$ is the differentiation matrix given in (4.1.27). When an explicit method is used to integrate (4.1.54) in time, the maximum allowable time step needs to satisfy

$$\Delta t = \mathcal{O}\left(\frac{1}{\nu\sigma(\widehat{D}^2)}\right), \tag{4.1.55}$$

where $\sigma(\hat{D}^2)$ is the spectral radii for $\hat{b}^2$ It can be shown (cf. [167]) that the spectral radii for the first and second Hermite differentiation matrices are $\mathcal{O}(\sqrt{N})$ and $\mathcal{O}(N)$, respectively. Hence, the stability condition (4.1.55) is very mild ($\Delta t \leqslant CN^{-1}$) compared to the Fourier case ($\Delta t \leqslant CN^{-2}$) and Legendre or Chebyshev cases ($\Delta t \leqslant CN^{-4}$). This very mild stability condition can be further alleviated by using a proper scaling. Indeed, since $\sigma(\widehat{D}^2) = \mathcal{O}(\alpha_N^2 N)$ with $N = \mathcal{O}(\sqrt{1/\nu})$ and $\alpha_N = \mathcal{O}(\sqrt{N})$ (see [167]), we obtain $\Delta t = \mathcal{O}(1)$. This suggests that the step-size in time can be independent of N when ν is small.

We now provide a pseudo-code for solving the heat equation (4.1.51) with the

Hermite spectral method. The ODE system (4.1.54) is integrated by using the forward Euler's method.

```
CODE Hermite*heat.1
Input N, ν, Δt, T, u0(x)
Call a subroutine to get the roots of H_{N+1}(x),i.e.,x_j,
0⩽j⩽N
Choose a scaling factor  α=x0/M
Call a subroutine to obtain Ĥ_{n,j}=Ĥ_n(x_j)
Compute the collocation point x(j) and the initial data:
 x(j)=x_j/α, u(j)=u0(x(j)), 0⩽j⩽N
time=0
While time < T
%form the Hermite expansion coefficient û_n
   for n=0, N do
      a(n)=Σ_{j=0}^{N} u(j)*Ĥ_n(x_j)
   endfor
%compute a2(n)=û_n^{(2)}
   a2(0)=0.5*α^2(-a(0)+√2 a(2)), a2(1)=0.5*α^2(-3*a(1)+√6 a(3))
   a2(N+1)=0,  a2(N+2)=0
   for n=2, N do
     a2(n)=0.5*α^2(√((n-1)n) a(n-2)-(2n+1)a(n)+√((n+1)(n+2)) a(n+2))
   endfor
 %forward Euler time integration
   for j=0 to N do
     u(j)=u(j)+ν*Δt*Σ_{n=0}^{N} a2(n)*Ĥ_n(x_j)
   endfor
 Update time:  time=time+Δt
endWhile
```

In the above code, a subroutine for finding the roots of $H_{N+1}(x)$ is similar to `CODE LGauss.1` given in Section 1.3. Moreover, the recurrence formula (4.1.14) must be used to evaluate $\hat{H}_n(x_j)$.

We use `CODE Hermite*heat.1` to solve the problem in Example 4.1.2. Figure 4.2 gives a comparison between the exact solution and the numerical results obtained by using a scaling factor, $\alpha = x_0/1.5$. The solution domain is therefore $|x| \leqslant 1.5$. With $N = 32$ and $\Delta t = 0.01$, it is seen that the agreement between the exact solution and the numerical result is good. However, if a scaling factor is not used, then reasonable approximate solutions cannot be obtained, even with a much larger values of N; see Figure 4.3.

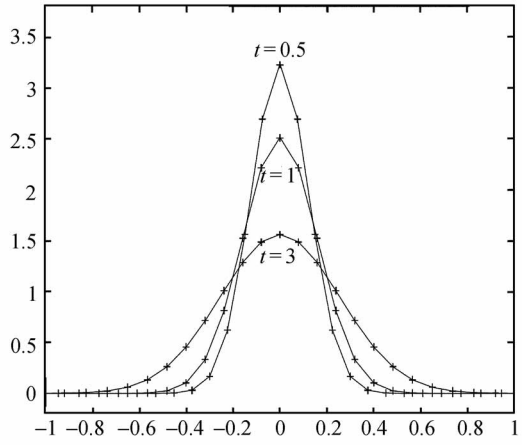

Figure 4.2

Example 4.1.2: Comparison between the exact (solid lines) and numerical (pluses) solutions at different time levels, with a constant scaling $\beta = x_0/1.5$, $N = 32$ and $\Delta t = 0.01$.

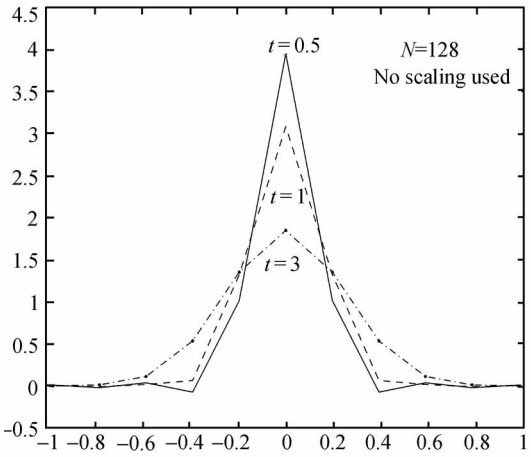

Figure 4.3

Example 4.1.2: Numerical solutions at different time levels without using a scaling (i.e., $\beta = 1$). $N = 128$ and $\Delta t = 0.005$. The exact solutions are displayed in Fig.4.2.

Example 4.1.3 *Consider the parabolic problem* (4.1.44) *with* $\nu = 1$ *and the following source term*

$$f(x,t) = \Big(x\cos x + (t+1)\sin x\Big)(t+1)^{-3/2}\exp\left(-\frac{x^2}{4(t+1)}\right). \qquad (4.1.56)$$

This example was proposed by Funaro and Kavian[54]. Its exact solution is of the form

$$u(x,t) = \frac{\sin x}{\sqrt{t+1}} \exp\left(-\frac{x^2}{4(t+1)}\right). \tag{4.1.57}$$

We denote

$$E_N(t) = \|u(t) - u_N(t)\|_{N,\omega_\alpha^{-1}}, \quad E_{N,\infty}(t) = \frac{\max_{0\leqslant j\leqslant N} |u_N(x_j,t) - u(x_j,t)|}{\max_{0\leqslant j\leqslant N} |u(x_j,t)|}.$$

We solve the above problem using (4.1.46) and the scaling factor (4.1.47) with $(\delta_0, \delta) = (0.5, 0)$ which corresponds to the classical approach, and with $(\delta_0, \delta) = (1, 1)$ which corresponds to a time-dependent scaling. For ease of comparison, we use the same mesh size as used in [54]. Table 4.1 shows the error $E_{20}(1)$ with different time steps. We note the result in [54] is obtained by an explicit first-order forward difference in time.

Table 4.2 shows the order of accuracy for the scheme (4.1.46) together with the Crank-Nicolson time-discretization. It is observed from the numerical results that the numerical scheme is of second-order accuracy in time and spectral accuracy in space.

Table 4.1 Errors at $t = 1$ with $N = 20$ using different methods

time step τ	Funaro and Kavian's scheme [54]	Classical method (4.1.46) with $(\delta_0, \delta) = (0.5, 0)$	Proposed method (4.1.46) with $(\delta_0, \delta) = (1, 1)$
250^{-1}	2.49E-03	1.95E-04	2.96E-06
1000^{-1}	6.20E-04	1.95E-04	1.19E-06
4000^{-1}	1.55E-04	1.95E-04	1.18E-06
16000^{-1}	3.89E-05	1.95E-04	1.18E-06

Table 4.2 Errors of the scheme (4.1.46) using the scaling factor (4.1.47) with $(\delta_0, \delta) = (1, 1)$

τ	N	$E_N(1)$	$E_{N,\infty}(1)$	Order
10^{-1}		1.70E-03	9.78E-04	
10^{-2}		1.70E-05	9.77E-06	$\tau^{2.00}$
10^{-3}	30	1.70E-07	9.77E-08	$\tau^{2.00}$
10^{-4}		1.70E-09	9.80E-10	$\tau^{2.00}$
	10	5.16E-03	1.19E-03	
10^{-4}	20	1.18E-06	1.25E-07	$N^{-12.10}$
	30	1.70E-09	9.80E-10	$N^{-16.14}$

Exercise 4.1

Problem 1 Approximate $f(x) = \sin(10x)$, $x \in [-\pi, \pi]$, using the Hermite spectral (or Hermite pseudo-spectral) method:

$$\sin(10x) \approx \sum_{n=0}^{N} a_n H_n(x). \tag{4.1.58}$$

a. Find a_n for $N = 11, 21$ and 31.

b. Let $x_j = 2\pi j/M$ ($j = -\frac{M}{2}, -\frac{M}{2} + 1, \cdots, \frac{M}{2}$) with $M = 200$. Plot the right-hand side function of (4.1.58) by using its values at $x = x_j$. Compare your results with the exact function $f(x) = \sin(10x)$.

c. Repeat (a) and (b) above by using an appropriate scaling factor, see (4.1.34).

Problem 2 Solve Example 4.1.1 to verify the claims for that example.

Problem 3 Solve problem (4.1.51) and (4.1.52) with $\nu = 0.01$. Compute $u(x, 2)$ by using the Hermite collocation method in space and RK4 in time with (i) $\Delta t = 0.1$, $N = 16$, (ii) $\Delta t = 0.01$, $N = 16$, (iii) $\Delta t = 0.01$, $N = 24$.

a. What are the good scaling factors for (i)–(iii) above?

b. Plot the exact solution and numerical solutions.

Problem 4 Repeat Problem 1 in Section 5.4 using a Hermite spectral method with a proper scaling.

4.2 Laguerre spectral methods

Generalized Laguerre polynomials
Laguerre functions
Interpolation, discrete transform and derivatives
Laguerre-collocation & Galerkin methods
General boundary conditions
Fourth-order equations
Scaling and numerical experiments

For problems posed on a semi-infinite interval, it is natural to consider the (usual) Laguerre polynomials $\{\mathcal{L}_n(x)\}$ which form a complete orthogonal systems in $L^2_\omega(0, \infty)$ with $\omega(x) = e^{-x}$. Although for many problems it is sufficient to use the usual Laguerre polynomials, it is more convenient for the analysis and for the implementation

to introduce a family of generalized Laguerre polynomials $\{\mathcal{L}_n^{(\alpha)}(x)\}$ with $\alpha > -1$ and $\mathcal{L}_n(x) = \mathcal{L}_n^{(0)}(x)$.

Generalized Laguerre polynomials

The generalized Laguerre polynomials $\mathcal{L}_n^{(\alpha)}(x)$ with $x \in \mathbb{R}_+ := (0, \infty), \alpha > -1$, are defined by the three-term recurrence relation:

$$\begin{aligned} &(n+1)\mathcal{L}_{n+1}^{(\alpha)}(x) = (2n+\alpha+1-x)\mathcal{L}_n^{(\alpha)}(x) - (n+\alpha)\mathcal{L}_{n-1}^{(\alpha)}(x), \\ &\mathcal{L}_0^{(\alpha)}(x) = 1, \quad \mathcal{L}_1^{(\alpha)}(x) = \alpha + 1 - x. \end{aligned} \tag{4.2.1}$$

They are orthogonal with respect to the weight function $\omega_\alpha(x) = x^\alpha e^{-x}$, i.e.,

$$\int_{\mathbb{R}_+} \mathcal{L}_n^{(\alpha)}(x)\mathcal{L}_m^{(\alpha)}(x)\omega_\alpha(x)\mathrm{d}x = \gamma_n^{(\alpha)}\delta_{mn}, \tag{4.2.2}$$

where

$$\gamma_n^{(\alpha)} = \Gamma(n+\alpha+1)/\Gamma(n+1). \tag{4.2.3}$$

We now collect some useful properties of the Laguerre polynomials/functions (cf. [155]):

a. Laguerre polynomials are eigenfunctions of the following Sturm-Liouville problem:

$$x^{-\alpha}e^x\partial_x\Big(x^{\alpha+1}e^{-x}\partial_x\mathcal{L}_n^{(\alpha)}(x)\Big) + \lambda_n\mathcal{L}_n^{(\alpha)}(x) = 0, \tag{4.2.4}$$

with the eigenvalues $\lambda_n = n$.

b. We derive from (4.2.4) the orthogonality for $\{\partial_x\mathcal{L}_n^{(\alpha)}(x)\}$:

$$\int_{\mathbb{R}_+} \partial_x\mathcal{L}_n^{(\alpha)}(x)\partial_x\mathcal{L}_m^{(\alpha)}(x)x\omega_\alpha(x)\mathrm{d}x = \lambda_n\gamma_n^{(\alpha)}\delta_{mn}. \tag{4.2.5}$$

c. Derivative relations:

$$\partial_x\mathcal{L}_n^{(\alpha)}(x) = -\mathcal{L}_{n-1}^{(\alpha+1)}(x) = -\sum_{k=0}^{n-1}\mathcal{L}_k^{(\alpha)}(x), \tag{4.2.6a}$$

$$\mathcal{L}_n^{(\alpha)}(x) = \partial_x\mathcal{L}_n^{(\alpha)}(x) - \partial_x\mathcal{L}_{n+1}^{(\alpha)}(x), \tag{4.2.6b}$$

$$x\partial_x\mathcal{L}_n^{(\alpha)}(x) = n\mathcal{L}_n^{(\alpha)}(x) - (n+\alpha)\mathcal{L}_{n-1}^{(\alpha)}(x). \tag{4.2.6c}$$

The two Gauss-type quadratures associated to (4.2.2) and (4.2.5) are:

- `Laguerre-Gauss quadrature` Let $\{x_j^{(\alpha)}\}_{j=0}^N$ be the zeros of $\mathcal{L}_{N+1}^{(\alpha)}(x)$

and define the associated weights by

$$\omega_j^{(\alpha)} = -\frac{\Gamma(N+\alpha+1)}{(N+1)!}\frac{1}{\mathcal{L}_N^{(\alpha)}(x_j^{(\alpha)})\partial_x\mathcal{L}_{N+1}^{(\alpha)}(x_j^{(\alpha)})}$$

$$= \frac{\Gamma(N+\alpha+1)}{(N+\alpha+1)(N+1)!}\frac{x_j^{(\alpha)}}{[\mathcal{L}_N^{(\alpha)}(x_j^{(\alpha)})]^2}, \qquad 0 \leqslant j \leqslant N. \tag{4.2.7}$$

Then we have

$$\int_{\mathbb{R}_+} p(x)x^\alpha e^{-x}\mathrm{d}x = \sum_{j=0}^{N} p(x_j^{(\alpha)})\omega_j^{(\alpha)}, \quad \forall p \in P_{2N+1}.$$

- `Laguerre-Gauss-Radau quadrature` Let $\{x_j^{(\alpha)}\}_{j=0}^N$ be the zeros of $x\partial_x\mathcal{L}_{N+1}^{(\alpha)}(x)$ and define the associated weights by

$$\omega_0^{(\alpha)} = \frac{(\alpha+1)\Gamma^2(\alpha+1)\Gamma(N+1)}{\Gamma(N+\alpha+2)}, \tag{4.2.8a}$$

$$\omega_j^{(\alpha)} = \frac{\Gamma(N+\alpha+1)}{N!(N+\alpha+1)}\frac{1}{[\mathcal{L}_N^{(\alpha)}(x_j^{(\alpha)})]^2}, \qquad 1 \leqslant j \leqslant N. \tag{4.2.8b}$$

Then we have

$$\int_{\mathbb{R}_+} p(x)x^\alpha e^{-x}\mathrm{d}x = \sum_{j=0}^{N} p(x_j^{(\alpha)})\omega_j^{(\alpha)}, \quad \forall p \in P_{2N}. \tag{4.2.9}$$

The zeros of $\mathcal{L}_{N+1}^{(\alpha)}(x)$ and $x\partial_x\mathcal{L}_{N+1}^{(\alpha)}(x)$ can be computed using Theorem 1.2.1.

The Laguerre-Gauss points (i.e., zeros of $\mathcal{L}_{N+1}^{(\alpha)}(x)$) can be computed as the eigenvalues of the symmetric tridiagonal matrix (1.2.5) with

$$\begin{aligned} &\alpha_j = 2j+\alpha+1, \quad 0 \leqslant j \leqslant N, \\ &\beta_j = j(j+\alpha), \quad 1 \leqslant j \leqslant N. \end{aligned} \tag{4.2.10}$$

Due to (4.2.6a), the Laguerre-Gauss-Radau points of order N with index α are simply the Laguerre-Gauss points of order $N-1$ with index $\alpha+1$ plus the point 0.

Let $\{x_j^{(\alpha,N)}\}_{j=0}^N$ be the Laguerre-Gauss-Radau points. It is well known (cf. [155]) that $x_N^{(\alpha,N)} \to +\infty$ as $N \to +\infty$. Hence, the values of $\mathcal{L}_N(x_N^{(\alpha,N)})$ grow extremely fast, and the associated weights in the Laguerre-Gauss-Radau quadrature decay extremely fast. Therefore, numerical procedures using Laguerre polynomials are usually very ill-conditioned. Instead, it is advisable to use Laguerre functions.

Laguerre functions

The generalized Laguerre function is defined by

$$\widehat{\mathcal{L}}_n^{(\alpha)}(x) := e^{-x/2}\mathcal{L}_n^{(\alpha)}(x), \quad \alpha > -1,\ x \in \mathbb{R}_+. \tag{4.2.11}$$

Let $\hat{\omega}_\alpha = x^\alpha$, we have that $\left\{\widehat{\mathcal{L}}_n^{(\alpha)}\right\}$ are $L^2_{\hat{\omega}_\alpha}(\mathbb{R}_+)-$orthogonal.

In what follows, we restrict ourselves to the most commonly used case,

$$\widehat{\mathcal{L}}_n(x) := \widehat{\mathcal{L}}_n^{(0)}(x) = e^{-x/2}\mathcal{L}_n(x), \ \ x \in \mathbb{R}_+, \tag{4.2.12}$$

which satisfies the three-term recurrence relation (derived from (4.2.1)):

$$\begin{aligned} &(n+1)\widehat{\mathcal{L}}_{n+1}(x) = (2n+1-x)\widehat{\mathcal{L}}_n(x) - n\widehat{\mathcal{L}}_{n-1}(x), \\ &\widehat{\mathcal{L}}_0(x) = e^{-x/2}, \quad \widehat{\mathcal{L}}_1(x) = (1-x)e^{-x/2}, \end{aligned} \tag{4.2.13}$$

and the orthogonality (cf. (4.2.2)):

$$\int_{\mathbb{R}_+} \widehat{\mathcal{L}}_n(x)\widehat{\mathcal{L}}_m(x)\mathrm{d}x = \delta_{mn}. \tag{4.2.14}$$

Moreover, due to the fact

$$\partial_x\widehat{\mathcal{L}}_n(x) = -\frac{1}{2}\widehat{\mathcal{L}}_n(x) + e^{-x/2}\partial_x\mathcal{L}_n(x), \tag{4.2.15}$$

we find from (4.2.6a) that

$$\partial_x\widehat{\mathcal{L}}_n(x) = -\sum_{k=0}^{n-1}\widehat{\mathcal{L}}_k(x) - \frac{1}{2}\widehat{\mathcal{L}}_n(x). \tag{4.2.16}$$

We emphasize that in contrast to Laguerre polynomials, the Laguerre functions

are well behaved, see Figure 4.4. More precisely, we have (cf. [155])

$$|\widehat{\mathcal{L}}_n(x)| \leqslant 1, \quad x \in \mathbb{R}_+; \tag{4.2.17a}$$

$$\widehat{\mathcal{L}}_n(x) = \mathcal{O}((nx)^{-1/4}), \quad \forall x \in [cn^{-1},\ \omega]. \tag{4.2.17b}$$

where ω is a finite real number.

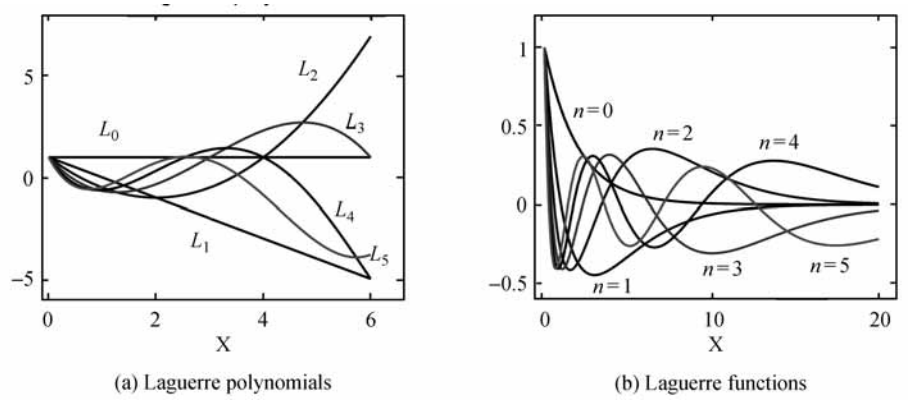

Figure 4.4

(a): Graphs of the first six Laguerre polynomials $\mathcal{L}_n(x)$ with $0 \leqslant n \leqslant 5$ and $x \in [0, 6]$; (b): Graphs of the first six Laguerre functions $\widehat{\mathcal{L}}_n(x)$ with $0 \leqslant n \leqslant 5$ and $x \in [0, 20]$.

It is straightforward to define Gauss-type quadrature rules associated with Laguerre function approach. As an example, we present below the Laguerre-Gauss-Radau quadrature associated with the Laguerre functions:

Let $\{x_j\}_{j=0}^N$ be the zeros of $x\partial_x \mathcal{L}_{N+1}(x)$, and

$$\hat{\omega}_j = \frac{1}{(N+1)[\widehat{\mathcal{L}}_N(x_j)]^2}, \quad 0 \leqslant j \leqslant N. \tag{4.2.18}$$

Then

$$\int_{\mathbb{R}_+} p(x)\mathrm{d}x = \sum_{j=0}^{N} p(x_j)\hat{\omega}_j, \quad \forall p \in \{u :\ u = e^{-x}v,\ v \in P_{2N}\}. \tag{4.2.19}$$

Interpolation, discrete transform and derivatives

We define

$$\widehat{P}_N = \text{span}\{\hat{\mathcal{L}}_k(x) : k = 0, 1, \cdots, N\}, \tag{4.2.20}$$

and denote by $\hat{I}_N$ the interpolation operator in $\widehat{P}_N$ based on the Laguerre-Gauss-Radau points $\{x_j\}_{j=0}^N$, i.e., for all $u \in C(\mathbb{R}_+)$, $\hat{I}_N$ satisfies

$$\hat{I}_N u(x_j) = u(x_j), \quad 0 \leqslant j \leqslant N. \tag{4.2.21}$$

For any $u \in \widehat{P}_N$, we write

$$u(x) = \sum_{n=0}^{N} \hat{u}_n \widehat{\mathcal{L}}_n(x), \quad u'(x) = \sum_{n=0}^{N} \hat{u}_n^{(1)} \widehat{\mathcal{L}}_n(x). \tag{4.2.22}$$

Thanks to the Laguerre-Gauss-Radau quadrature, the interpolation coefficients $\{\hat{u}_n\}$ can be determined by

$$\hat{u}_n = \frac{1}{N+1} \sum_{j=0}^{N} \frac{\widehat{\mathcal{L}}_n(x_j)}{[\widehat{\mathcal{L}}_N(x_j)]^2} u(x_j), \quad 0 \leqslant n \leqslant N. \tag{4.2.23}$$

By using the recurrence relation (4.2.16), we can compute $\{\hat{u}_n^{(1)}\}_{n=0}^N$ from the coefficients $\{\hat{u}_n\}_{n=0}^N$ as follows:

$$\begin{aligned} \hat{u}_N^{(1)} &= -\frac{1}{2}\hat{u}_N, \\ \hat{u}_k^{(1)} &= -\frac{1}{2}\hat{u}_k - \sum_{n=k+1}^{N} \hat{u}_n, \quad 0 \leqslant k \leqslant N-1. \end{aligned} \tag{4.2.24}$$

We now turn to the differentiation matrix corresponding to the Laguerre function approach. Let $\{x_j\}_{j=0}^N$ be the Laguerre-Gauss-Radau points. Given $u \in \widehat{P}_N$, we can write

$$u(x) = \sum_{j=0}^{N} u(x_j) \hat{h}_j(x), \tag{4.2.25}$$

where $\{\hat{h}_j\}_{j=0}^N$ are the Lagrange interpolation functions satisfying

$$\hat{h}_j \in \widehat{P}_N, \quad \hat{h}_j(x_k) = \delta_{kj}, \quad 0 \leqslant k, j \leqslant N.$$

It can be verified that the first-order differentiation matrix associated with the Laguerre-Gauss-Radau points is given by

$$\hat{d}_{kj} = \hat{h}_j'(x_k) = \begin{cases} \dfrac{\widehat{\mathcal{L}}_{N+1}(x_k)}{(x_k - x_j)\widehat{\mathcal{L}}_{N+1}(x_j)}, & k \neq j, \\ 0, & k = j \neq 0, \\ -(N+1)/2, & k = j = 0. \end{cases} \tag{4.2.26}$$

Laguerre-collocation & Galerkin methods

To illustrate how to solve differential equations in the semi-infinite interval using Laguerre functions, we consider the following model equation:

$$\begin{aligned} &-u''(x)+\gamma u(x)=f(x), \quad x\in\mathbb{R}_+, \ \ \gamma>0,\\ &u(0)=0, \quad \lim_{x\to+\infty}u(x)=0. \end{aligned} \tag{4.2.27}$$

`Laguerre-collocation method` The Laguerre-collocation method for (4.2.27) is to find $u_N\in\hat{P}_N$ such that

$$\begin{aligned} &-u_N''(x_j)+\gamma u_N(x_j)=f(x_j), \quad 1\leqslant j\leqslant N,\\ &u_N(0)=0. \end{aligned} \tag{4.2.28}$$

Setting $\bar{u}=(u_N(x_0),u_N(x_1),\cdots,u_N(x_N))^{\mathrm{T}}$ and $\bar{f}=(f(x_0),f(x_1),\cdots,f(x_N))^{\mathrm{T}}$, the collocation equation (4.2.28) can be written in the form

$$(-\widehat{D}^2+\gamma I)\bar{u}=\bar{f}, \tag{4.2.29}$$

where $\widehat{D}=(\hat{d}_{kj})$ is the $N\times N$ matrix with $\hat{d}_{kj}$ defined in (4.2.26).

`Laguerre-Galerkin methods` We now consider the approximation of (4.2.27) by using a Galerkin method. To this end, we define

$$H_0^1(\mathbb{R}_+)=\{u\in H^1(\mathbb{R}_+)\ :\ u(0)=0\}, \quad \widehat{P}_N^0=\{u\in\widehat{P}_N:\ u(0)=0\}.$$

Then, the variational formulation of (4.2.27) is to find $u\in H_0^1(\mathbb{R}_+)$ such that

$$a(u,v):=(u',v')+\gamma(u,v)=(f,v), \quad \forall v\in H_0^1(\mathbb{R}_+). \tag{4.2.30}$$

The Laguerre-Galerkin approximation $u_N\in\widehat{P}_N^0$ to (4.2.30) is determined by

$$a(u_N,v_N)=(u_N',v_N')+\gamma(u_N,v_N)=(\hat{I}_Nf,v_N), \quad \forall v_N\in\widehat{P}_N^0, \tag{4.2.31}$$

where $\hat{I}_Nf\in\widehat{P}_N$ is the interpolating function such that $\hat{I}_Nf(x_i)=f(x_i),\ i=0,1,\cdots,N$.

Let us set

$$\hat{\phi}_k(x)=(\mathcal{L}_k(x)-\mathcal{L}_{k+1}(x))e^{-x/2}=\widehat{\mathcal{L}}_k(x)-\widehat{\mathcal{L}}_{k+1}(x). \tag{4.2.32}$$

It follows from $\mathcal{L}_k(0) = 1$ that $\hat{\phi}_k(0) = 0$, and

$$\widehat{P}_N^0 = \text{span}\{\hat{\phi}_0, \hat{\phi}_1, \cdots, \hat{\phi}_{N-1}\}. \tag{4.2.33}$$

Hence, defining

$$\begin{aligned}
&u_N = \sum_{k=0}^{N-1} \hat{u}_k \hat{\phi}_k(x), \quad \bar{u} = (\hat{u}_0, \hat{u}_1, \cdots, \hat{u}_{N-1})^{\mathrm{T}},\\
&f_i = (\hat{I}_N f, \hat{\phi}_i), \quad \bar{f} = (f_0, f_1, \cdots, f_{N-1})^{\mathrm{T}},\\
&s_{ik} = (\hat{\phi}_k', \ \hat{\phi}_i'), \quad S = (s_{ik})_{0\leqslant i,k\leqslant N-1},\\
&m_{ik} = (\hat{\phi}_k, \ \hat{\phi}_i), \quad M = (m_{ik})_{0\leqslant i,k\leqslant N-1},
\end{aligned}$$

we see that M is a symmetric tridiagonal matrix, and it can be verified that $S = I - \frac{1}{4}M$. Thus, the system (4.2.31) is reduced to the matrix form

$$\Big(I + (\gamma - 1/4)M\Big)\bar{u} = \bar{f}. \tag{4.2.34}$$

General boundary conditions

More general boundary conditions of the form

$$au(0) - bu'(0) = c, \tag{4.2.35}$$

with $b \neq 0$ ($b = 0$ reduces to the simpler Dirichlet case) and $ab \geqslant 0$ to ensure the ellipticity, can be easily handled as follows. First, the non-homogeneity can be taken care of by subtracting $u_b(x) := c/(a + b/2)e^{-x/2}$ from the solution, leading to the following homogeneous problem: Find $u = \tilde{u} + u_b$ such that

$$\gamma\tilde{u} - \tilde{u}_{xx} = f - (\gamma - 1/4)u_b := \tilde{f}; \quad a\tilde{u}(0) - b\tilde{u}'(0) = 0. \tag{4.2.36}$$

The corresponding variational formulation is: Find $u = \tilde{u} + u_b \in H^1(\mathbb{R}_+)$ such that

$$\gamma(\tilde{u}, v) + \frac{a}{b}\tilde{u}(0)v(0) + (\tilde{u}_x, v_x) = (\tilde{f}, v), \quad \forall v \in H^1(\mathbb{R}_+). \tag{4.2.37}$$

Since $\mathcal{L}_k(0) = 1$ and $\partial_x\mathcal{L}_k(0) = -k$, we find that

$$\begin{aligned}
\tilde{\phi}_k(x) &= (\mathcal{L}_k(x) - a_k\mathcal{L}_{k+1}(x))e^{-x/2},\\
a_k &= (a + kb + b/2)/(a + (k + 1)b + b/2),
\end{aligned} \tag{4.2.38}$$

satisfies $a\tilde{\phi}_k(0) - b\tilde{\phi}_k'(0) = 0$. Let

$$\widetilde{X}_N = \text{span}\{\tilde{\phi}_0, \tilde{\phi}_1, \cdots, \tilde{\phi}_{N-1}\}. \tag{4.2.39}$$

Thus, the Laguerre-Galerkin method for (4.2.37) is: Find $u_N = \tilde{u}_N + u_b$ with $\tilde{u}_N \in \widetilde{X}_N$ such that

$$\gamma(\tilde{u}_N, \tilde{v}) + (\tilde{u}_N', \tilde{v}') + \frac{a}{b}\tilde{u}_N(0)\tilde{v}(0) = (\hat{I}_N\tilde{f}, \tilde{v}), \quad \forall \tilde{v} \in \widetilde{X}_N. \tag{4.2.40}$$

It is clear that $\tilde{M}_{ij} := (\tilde{\phi}_j, \tilde{\phi}_i) = 0$ for $|i-j| > 1$. One also verifies readily by integration by parts that

$$\tilde{S}_{ij} := (\tilde{\phi}_j', \tilde{\phi}_i') + \frac{a}{b}\tilde{\phi}_j(0)\tilde{\phi}_i(0) = -(\tilde{\phi}_j'', \tilde{\phi}_i) = -(\tilde{\phi}_j, \tilde{\phi}_i''), \tag{4.2.41}$$

which implies that $\tilde{S}_{ij} = 0$ for $|i-j| > 1$. Hence, the matrix $\tilde{S} + \alpha\tilde{M}$ associated to (4.2.40) is again tridiagonal.

Fourth-order equations

Consider the fourth-order model problem

$$\begin{aligned} &\alpha_1 u - \alpha_2 u_{xx} + u_{xxxx} = f, \ \ x \in \mathbb{R}_+ \\ &u(0) = u_x(0) = \lim_{x\to+\infty} u(x) = \lim_{x\to+\infty} u_x(x) = 0. \end{aligned} \tag{4.2.42}$$

Let $H_0^2(\mathbb{R}_+) = \{u \in H^2(\mathbb{R}_+) : u(0) = u_x(0) = 0\}$. The variational formulation for (4.2.42) is: Find $u \in H_0^2(\mathbb{R}_+)$ such that

$$\alpha_1(u, v) + \alpha_2(u_x, v_x) + (u_{xx}, v_{xx}) = (f, v), \quad \forall v \in H_0^2(\mathbb{R}_+). \tag{4.2.43}$$

Set $\widehat{X}_N = \{u \in \widehat{P}_N : u(0) = u_x(0) = 0\}$. Then, the Laguerre-Galerkin approximation for (4.2.43) is: Find $u_N \in \widehat{X}_N$ such that

$$\alpha_1(u_N, v) + \alpha_2(u_N', v') + (u_N'', v'') = (\hat{I}_N f, v), \quad \forall v \in \widehat{X}_N. \tag{4.2.44}$$

One verifies easily that

$$\hat{\psi}_k(x) = (\mathcal{L}_k(x) - 2\mathcal{L}_{k+1}(x) + \mathcal{L}_{k+2}(x))e^{-x/2} \in \widehat{X}_{k+2},$$

and $\widehat{X}_N = \text{span}\{\hat{\psi}_0, \hat{\psi}_1, \cdots, \hat{\psi}_{N-2}\}$. Hence, setting

$$\begin{aligned} &s_{kj} = (\hat{\psi}_j'', \hat{\psi}_k''), \quad S = (s_{kj})_{k,j=0,1,\cdots,N-2}, \\ &q_{kj} = (\hat{\psi}_j', \hat{\psi}_k'), \quad Q = (q_{kj})_{k,j=0,1,\cdots,N-2}, \\ &m_{kj} = (\hat{\psi}_j, \hat{\psi}_k), \quad M = (m_{kj})_{k,j=0,1,\cdots,N-2}, \\ &\tilde{f}_k = (\hat{I}_N f, \hat{\psi}_k), \quad \bar{f} = (\tilde{f}_0, \tilde{f}_1, \cdots, \tilde{f}_{N-2}), \\ &u_N = \sum_{k=0}^{N-2} \tilde{u}_k \hat{\psi}_k, \quad \bar{u} = (\tilde{u}_0, \tilde{u}_1, \cdots, \tilde{u}_{N-2}), \end{aligned} \tag{4.2.45}$$

we find that (4.2.44) reduces to

$$(\alpha_1 M + \alpha_2 Q + S)\bar{u} = \bar{f}. \tag{4.2.46}$$

One verifies that S, Q and M are all symmetric penta-diagonal and their entries can be easily computed. Hence, (4.2.46) can be efficiently solved.

Scaling and numerical experiments

Although the Laguerre-spectral methods presented above enjoy a theoretical spectral convergence rate, the actual error decays considerably slower than that of the Chebyshev- or Legendre-spectral method for similar problems in finite intervals. The poor resolution property of Laguerre polynomials/functions, which was pointed out by Gottlieb and Orszag in [61], is one of the main reasons why Laguerre polynomials/functions are rarely used in practice. However, similarly as in [158] for the Hermite spectral method, the resolution of Laguerre functions can be greatly improved by using a proper scaling factor.

The main factor responsible for the poor resolution of Laguerre polynomials and Laguerre functions is that usually a significant portion of the Laguerre Gauss-Radau points is located outside of the interval of interest. For example, $u(x) = (\sin kx)e^{-x} \leqslant 10^{-8}$ for $x > 18$, so all the collocation points which are greater than 18 are essentially wasted. Thus, it makes sense to scale the function so that all the effective collocation points are inside the interval of interest. More precisely, we can proceed as follows: Given an accuracy threshold ε, we estimate a M such that $|u(x)| \leqslant \varepsilon$ for $x > M$. Then we set the scaling factor $\beta_N = x_N^{(N)}/M$, where $x_N^{(N)}$ is the largest Laguerre Gauss-Lobatto point, and instead of solving the equation (4.2.27), we solve the following scaled equation with the new variable $y = \beta_N x$:

$$\gamma v - \beta_N^2 v_{yy} = g(y);\ v(0) = 0,\ \lim_{y\to+\infty} u(y) = 0, \tag{4.2.47}$$

where $v(y) = u(\beta_N x)$ and $g(y) = f(\beta_N x)$. Thus, the effective collocation points $x_j = y_j/\beta_N$ ($\{y_j\}_{j=0}^N$ being the Laguerre Gauss-Lobatto points) are all located in $[0, M]$. In Figure 4.5, the approximations of (4.2.27) with the exact solution being $u(x) = \sin 10x(x+1)^{-5}$ using the Laguerre-Galerkin method with a scaling factor=15 and without scaling are plotted against the exact solution. Notice that if no scaling is used, the approximation with $N = 128$ still exhibits an observable error, while the approximation with a scaling factor=15 using only 32 modes is virtually indistinguishable with the exact solution. This simple example demonstrates that a proper scaling will greatly enhance the resolution capabilities of the Laguerre functions and make the Laguerre functions a viable alternative to the rational polynomials studied in [66], [18].

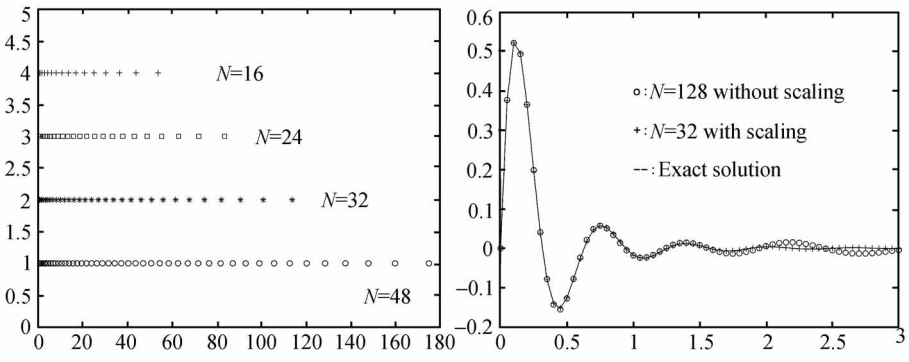

Figure 4.5 Locations of the Laguerre Gauss-Radau points and effects of scaling

Example 4.2.1 *The Schrödinger equation,*

$$-y''(x) + y(x) = \lambda q(x) y(x), \qquad 0 < x < \infty, \tag{4.2.48}$$

plays a central role in the theory of Quantum Mechanics. Here,

$$q(x) = \frac{1}{1 + \mathrm{e}^{(x-r)/\epsilon}},$$

with $r = 5.08685476$ *and* $\epsilon = 0.929852862$. *The boundary conditions are*

$$y(0) = 0, \quad \lim_{x\to\infty} y(x) = 0. \tag{4.2.49}$$

This problem can be solved by using Weideman and Reddy's MATLAB Differentiation Matrix Suite (cf. [168]). Since the domain is $[0, \infty)$, solving the Schrödinger

equation by the Laguerre spectral collocation method is a natural choice. Let D be the second-derivative Laguerre matrix of order $N+1$, as computed by `lagdif.m` in [168]. Let the scaling parameter be β. This means that the nodes are $x_j = r_j/\beta$, where the r_j are the roots of $xL'_{N+1}(x)$. There is an additional boundary node $x = 0$; incorporation of the boundary condition at this node means the first row and column of D are to be deleted. The boundary condition at $x = \infty$ is automatically taken care of by the Laguerre expansion. The Schrödinger equation is therefore approximated by the $N \times N$ matrix eigenvalue problem

$$(-D + I)\mathbf{y} = \lambda Q\mathbf{y}, \tag{4.2.50}$$

where $\mathbf{y}$ represents the approximate eigenfunction values at the nodes, I is the identity matrix, and

$$Q = \mathrm{diag}\left(\frac{1}{1 + \mathrm{e}^{(x_j - r)/\epsilon}}\right).$$

The MATLAB function `schrod.m` in [168] given in Table 4.3 implements this method.

The physically interesting eigenvalue is the one of smallest magnitude. It was computed before to seven-digit accuracy as $\lambda = 1.424333$. The Laguerre method shown in Table 4.3 computed this eigenvalue to full accuracy with $N = 20$ (resp. $N = 30$) and all scaling parameters roughly in the range $\beta \in [3, 6]$ (resp. $\beta \in [2, 9]$).

Table 4.3 Computing the smallest eigenvalue of the Schrödinger Equation

```
>>b = 4; N = 20;                          % Initialize parameters.
>>r = 5.08685476; epsi = 0.929852862;

>>[x,D] = lagdif(N+1,2,b);                % Compute Laguerre derivative
                                           matrix.
>>D2=D(2:N+1,2:N+1);

>>Q = diag(1./(1+exp((x-r)/epsi)));       % Woods-Saxon potential.
>>I = eye(size(D2));                      % Identity matrix.

>>e = min(eig(-D2+I,Q));                  % Compute smallest eigenvalue.
```

Exercise 4.2

Problem 1 Solve the boundary value problem

$$u''(y) - y^2 u(y) = -e^{-y^2/2}, \qquad y \in (1, \infty)$$
$$u(1) = e^{-1/2},$$

by using the Laguerre-spectral method.

Problem 2 Solve the problem

$$\frac{\partial u}{\partial t} - \nu \frac{\partial^2 u}{\partial x^2} = 0, \qquad x \in (0, \infty),$$

$$u(x,0) = \frac{1}{\sqrt{\nu\pi}} \exp\left(-\frac{x^2}{\nu}\right), \qquad u(0,t) = \frac{1}{\sqrt{\pi\nu(4t+1)}},$$

with $\nu = 0.01$ by using the Laguerre-collocation method in space and RK4 in time with (i) $\Delta t = 0.1$, $N = 16$; (ii) $\Delta t = 0.01$, $N = 16$; (iii) $\Delta t = 0.01$, $N = 24$. The exact solution is given by (4.1.53).

a. What are the good scaling factors for (i)–(iii) above?

b. Plot the exact solution and numerical solutions.

4.3 Spectral methods using rational functions

Rational spectral method in the whole line
Rational spectral method in a semi-infinite interval
Numerical experiments

In this section, we will discuss the use of rational spectral methods, which are particularly suitable for problems whose solutions do not decay exponentially to zero as $|x| \to \infty$. The properties of the rational spectral methods have been discussed by several researchers, see, e.g. Boyd [17, 19] and Weideman [167].

Rational spectral method in the whole line

For problem posed in the whole line, a suitable set of rational basis functions is defined by

$$R_n(t) = \cos(n\cot^{-1}(t)), \qquad n = 0, 1, \cdots . \tag{4.3.1}$$

The above orthogonal rational functions are merely mapped Chebyshev polynomials,which in turn are the transformed cosines of a Fourier series. With the map $x = t/\sqrt{1+t^2}$, the basis functions defined by (4.3.1) are equal to $T_n(x)$, where the $T_n(x)$ are the usual Chebyshev polynomials. The first five basis functions are

$$R_0(t) = 1, \quad R_1(t) = \frac{t}{\sqrt{t^2+1}}, \quad R_2(t) = \frac{t^2-1}{t^2+1},$$

$$R_3(t) = \frac{t(t^2-3)}{(t^2+1)^{3/2}}, \quad R_4(t) = \frac{t^4-6t^2+1}{(t^2+1)^2}. \tag{4.3.2}$$

In general, only the R_n's with even n are truly rational but the others have a square root in the denominator. The orthogonality relation is

$$\int_{-\infty}^{\infty} \frac{1}{1+t^2} R_m(t) R_n(t) \mathrm{d}t = \frac{\pi c_n}{2} \delta_{m,n}, \tag{4.3.3}$$

where $c_0 = 2, c_n = 1 (n \geqslant 1)$ and $\delta_{m,n}$ is the Kronecker–delta. Thus, if $f(t) \in L^2(\mathbb{R})$ and $f(t) = \sum_{n=0}^{\infty} a_n R_n(t)$, then we have

$$a_n = \frac{2}{\pi c_n} \int_{-\infty}^{\infty} \frac{1}{1+t^2} f(t) R_n(t) \mathrm{d}t, \qquad n \geqslant 0.$$

Example 4.3.1 *As an example, we consider parametrized dynamical systems of the form*

$$u' = f(u, \lambda), \qquad u(t) \in \mathbb{R}^d, \quad \lambda \in \mathbb{R}^p, \quad t \in \mathbb{R}, \tag{4.3.4}$$

where $d, p \geqslant 1$. A solution $u(t)$ of (4.3.4) *at λ is called a connecting orbit if the limits*

$$u_- = \lim_{t \to -\infty} u(t), \qquad u_+ = \lim_{t \to \infty} u(t) \tag{4.3.5}$$

exist.

In the case $u_- = u_+$, the orbit is called a homoclinic orbit; when $u_- \neq u_+$, it is called a heteroclinic orbit. A closed path formed by several heteroclinic orbits is called a heteroclinic cycle. Homoclinic orbits typically arise as limiting cases of periodic orbits which attain infinite period but stay bounded in phase space. There are also many applications for studying the heteroclinic orbits. For example, the problem of finding traveling wave front solutions of constant speed for nonlinear parabolic equations is equivalent to the problem of finding trajectories that connect two fixed points of an associated system of ordinary differential equations (ODEs). Computation of connecting orbits involves the solution of a boundary value problem on the real line. Therefore, the problem is frequently replaced by one on a finite domain. The system of ODEs is then solved by a standard ODE boundary value solvers such as a multiple shooting methods and spline collocation methods. A more efficient method is to employ the rational spectral method. This procedure does not require that the infinite interval be truncated. Furthermore, spectral accuracy can be expected with this approach. Accurate numerical results can be obtained using a small number of grid points.

We now consider the use of the rational spectral method to solve (4.3.4). Let $u = (u_1, \cdots, u_d)^{\mathrm{T}}$ and $f = (f_1, \cdots, f_d)^{\mathrm{T}}$. Substituting the expansions

$$u_i(t) = \sum_{k=0}^{M+1} c_{ik} R_k(t), \qquad 1 \leqslant i \leqslant d, \tag{4.3.6}$$

into (4.3.4), we obtain

$$\sum_{k=0}^{M+1} c_{ik} R_k'(t) = f_i \left(\left(\sum_{k=0}^{M+1} c_{1k} R_k(t), \cdots, \sum_{k=0}^{M+1} c_{dk} R_k(t) \right)^{\mathrm{T}}, \lambda \right), \quad 1 \leqslant i \leqslant d, \tag{4.3.7}$$

where M is a given positive integer. The derivatives of $R(t)$, $R'(t)$, can be obtained by direct calculations from (4.3.1).

In practical calculations, it is more efficient to use the pseudospectral method. That is, we assume (4.3.7) hold at the collocation points $\{t_j\}_{j=1}^{M}$. As mentioned before, our basis functions $R_n(t)$ are mapped Chebyshev polynomials,which suggests to choose the collocation points as

$$t_j = \cot\left(j\pi/(M+1)\right), \qquad 1 \leqslant j \leqslant M. \tag{4.3.8}$$

Due to the nature of the rational spectral functions we can add two collocation points, $t_0 = +\infty$ and $t_{M+1} = -\infty$. Using the relation

$$u_i(t_j) = \sum_{k=0}^{M+1} c_{ik} \cos\left(kj\pi/(M+1)\right), \quad 0 \leqslant j \leqslant M+1, \tag{4.3.9}$$

we obtain

$$c_{ik} = \frac{2}{(M+1)\bar{c}_k} \sum_{m=0}^{M+1} \bar{c}_m^{-1} u_i(t_m) \cos\left(mk\pi/(M+1)\right), \quad 0 \leqslant k \leqslant M+1, \tag{4.3.10}$$

where $\bar{c}_m = 2$ if $m = 0$ or $M+1$, and $\bar{c}_m = 1$ if $1 \leqslant m \leqslant M$. Using (4.3.10) and

$$R_k'(t_j) = k \sin^2\left(j\pi/(M+1)\right) \sin\left(kj\pi/(M+1)\right), \tag{4.3.11}$$

we have, for $1 \leqslant j \leqslant M$,

$$u_i'(t_j) = \sum_{k=0}^{M+1} c_{ik} R_k'(t_j) = \sum_{k=0}^{M+1} c_{ik} k \sin^2\left(j\pi/(M+1)\right) \sin\left(kj\pi/(M+1)\right)$$

$$= \frac{2}{M+1}\sin^2(j\pi/(M+1))\sum_{k,m}\frac{k}{\bar{c}_k\bar{c}_m} \cos(mk\pi/(M+1))\sin(kj\pi/(M+1))\,u_i(t_m).$$

The problem (4.3.4) and (4.3.5) is to solve

$$\begin{aligned} &\frac{2}{M+1}\sin^2(j\pi/(M+1))\sum_{k,m}\frac{k}{\bar{c}_k\bar{c}_m}\cos(mk\pi/(M+1)) \\ &\sin(kj\pi/(M+1))\,u_i(t_m) \\ &= f_i(u(t_j),\lambda), \qquad 1\leqslant i\leqslant d, \quad 1\leqslant j\leqslant M, \\ &u(t_0)=u_+, u(t_{M+1})=u_-. \end{aligned} \tag{4.3.12}$$

The above system has dM equations for the dM unknowns $u_i(t_j), 1\leqslant i\leqslant d, 1\leqslant j\leqslant M$. The main advantage of the pseudospectral method is that it allows one to work in the physical space rather than the coefficient space. Thus it is possible to handle nonlinearities very efficiently, without the convolution sums introduced by the pure spectral method.

We can also solve the problem (4.3.4) and (4.3.5) by using the collocation points (4.3.8) in the equation (4.3.7). It follows from (4.3.11) and $R_k(t_j)=\cos(kj\pi/(M+1))$ that

$$\begin{aligned} &\sum_{k=0}^{M+1} k\sin^2(j\pi/(M+1))\sin(kj\pi/(M+1))\,c_{ik} \\ =&f_i\left(\left(\sum_{k=0}^{M+1}c_{1k}\cos(kj\pi/(M+1)),\cdots,\sum_{k=0}^{M+1}c_{dk}\cos(kj\pi/(M+1))\right)^{\mathrm{T}},\lambda\right), \end{aligned} \tag{4.3.13}$$

for $1\leqslant j\leqslant M, 1\leqslant i\leqslant d$. The above system gives dM equations. We can use further $2d$ conditions, which are given by (4.3.5), to find the $d(M+2)$ unknowns $c_{ik}, 1\leqslant i\leqslant d, 0\leqslant k\leqslant M+1$.

Rational spectral method in a semi-infinite interval

By applying a mapping to the Chebyshev polynomial,we define a new spectral basis[20, 22]. The new basis functions, denoted by $TL_n(y)$, are defined by

$$TL_n(y) := T_n(x) = \cos(nt), \tag{4.3.14}$$

where L is a constant map parameter and the three coordinates are related by

$$y = L\frac{1+x}{1-x}, \qquad x = \frac{y-L}{y+L}, \tag{4.3.15}$$

$$y = L\cot^2\frac{t}{2}, \qquad t = 2\cot^{-1}\sqrt{\frac{y}{L}}. \tag{4.3.16}$$

To avoid confusion as we leap from one coordinate to another, we shall adopt the convention that $y \in [0,\infty)$ is the argument of the $TL_n(y)$, $x \in [-1,1]$ is the argument of the ordinary Chebyshev polynomials,and $t \in [0,\pi]$ is the argument for cosines. We are free to calculate in whichever of these three coordinates are most convenient.

We shall refer to the $TL_n(y)$ as the *rational Chebyshev functions on a semi-infinite interval*. The first five basis functions for $L = 1$ are

$$TL_0(y) = 1, \quad TL_1(y) = \frac{y-1}{y+1}, \quad TL_2(y) = \frac{y^2-6y+1}{(y+1)^2},$$
$$TL_3(y) = \frac{y^3-15y^2+15y-1}{(y+1)^3}, \quad TL_4(y) = \frac{y^4-28y^3+70y^2-28y+1}{(y+1)^4}. \tag{4.3.17}$$

By merely changing the variable in the usual orthogonality integral for the cosines, one can show that the rational Chebyshev functions are orthogonal:

$$\int_0^\infty \frac{TL_m(y)TL_n(y)\sqrt{L}}{\sqrt{y}(y+L)}\mathrm{d}y = \frac{\pi c_n}{2}\delta_{m,n}, \tag{4.3.18}$$

where $c_0 = 2, c_n = 1\,(n \geqslant 1)$. The pseudospectral grid in y is simply the image under the mapping of an evenly spaced Fourier grid,

$$y_i = L\cot^2\frac{t_i}{2}, \quad t_i = \frac{\pi(2i+1)}{2N+2}, \quad 0 \leqslant i \leqslant N. \tag{4.3.19}$$

If we have a differential equation defined in the interval $[\alpha,\infty)$ instead of $y \in [0,\infty)$, then we merely generalize (4.3.16) to $y = \alpha + L\cot^2(t/2)$. The relevant modification is that the collocation grid is changed to

$$y_i = \alpha + L\cot^2\frac{t_i}{2}, \quad t_i = \frac{\pi(2i+1)}{2N+1}, \qquad 0 \leqslant i \leqslant N-1. \tag{4.3.20}$$

A boundary condition is to be imposed at y_N (note that $TL_n(\alpha) = TL_n(y_N) = \cos(n\pi) = (-1)^n$).

Numerical experiments

Example 4.3.2 *The Huxley equation,*

$$\begin{aligned} &w_t = w_{zz} + f(w,a), \quad x \in \mathbb{R}, \quad t > 0, \\ &\quad f(w,a) := w(1-w)(w-a), \quad a \in (0,1). \end{aligned} \tag{4.3.21}$$

We look for traveling wave solutions to (4.3.21) *of the form* $w(z,t) = u(z+bt)$*, where* b *is the wave speed.*

The problem (4.3.21) gives the first-order ODE system

$$\begin{aligned} \frac{du_1(x)}{dx} &= u_2(x), \\ \frac{du_2(x)}{dx} &= bu_2(x) - f(u_1(x),a), \end{aligned} \tag{4.3.22}$$

where $x = z + bt$. If $a = 0.5$ and $b = 0$, then (4.3.22) has a family of periodic orbits of increasing period. In the limit, as the period goes to infinity, the orbits approach a heteroclinic cycle with the equilibrium points $(0,0)$ and $(1,0)$. The first of the two heteroclinic orbits has the exact representation

$$u_1(x) = \frac{\exp(x/\sqrt{2})}{1+\exp(x/\sqrt{2})}, \quad u_2(x) = \frac{du_1(x)}{dx}. \tag{4.3.23}$$

The second heteroclinic connection is obtained by reflecting the phase plane representation of the first with respect to the horizontal axis $u_2 = 0$. Since this test problem has (4.3.23) as an exact solution, it is useful to test our spectral method.

It is known that a scaling factor β is useful to optimize computational accuracy. It is observed in [105] that if β is not carefully chosen, then numerical oscillations are present. It is also observed that a reasonable choice of β in this case is in the region $[0.1, 0.5]$, since this leads to smooth curves. It was found that the accuracy is not very sensitive to the choice of the scaling factor in the neighborhood of the optimum β. Therefore, it is safe to use any β in this "trusted" region. Based on this observation, for any given M we can obtain a corresponding interval from which we can choose any value as β.

For Example 4.3.2, the exact representation of the two branches is $b = \pm\sqrt{2}(a - 0.5)$. Therefore in the case $a = 0.5$ the exact value of b is 0. In [105], the numerical values of $|b|$ against the number of collocation points, M, are presented, with corresponding scaling factors used. It was observed that if $M > 10$, then the scaling

factors used are almost independent of M. For a given M, we define the numerical errors as

$$\text{error} \equiv \max_{1 \leqslant j \leqslant M} \|u(t_j) - U(t_j)\|_{l^2}, \tag{4.3.24}$$

where $u = (u_1, u_2)^{\mathrm{T}}$ is the exact solution given in (4.3.23), and U is the numerical solution. The numerical errors are plotted in Fig. 4.6, which also shows a spectral convergence rate.

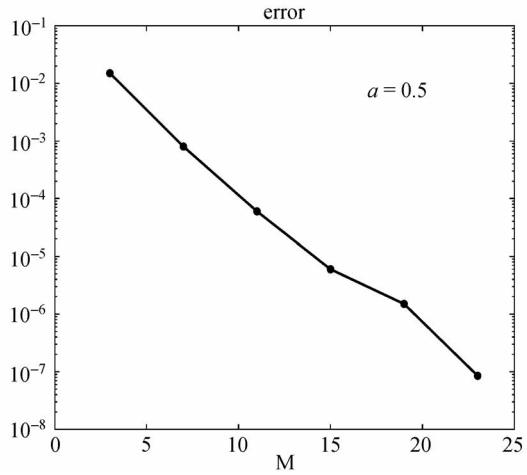

Figure 4.6 The maximum errors between the exact and the numerical solutions of u_1.

Exercise 4.3

Problem 1 The differential equation

$$u''(y) - y^2 u(y) = -e^{-y^2/2}, \qquad y \in \mathbb{R}$$

has an exact solution $u(y) = e^{-y^2/2}$. Solve it by using the rational spectral method. Show spectral accuracy by using a number of collocation points.

Problem 2 Solve

$$\begin{aligned} &u''(y) - y^2 u(y) = -e^{-y^2/2}, \qquad y \in (1, \infty), \\ &u(1) = e^{-1/2}, \end{aligned}$$

by using the rational spectral method.

4.4 Error estimates in unbounded domains

Laguerre-Galerkin method
Hermite-Galerkin method

In this section, we will discuss some basic technqiues for obtaining error bounds for spectral methods in unbounded domains. There have been several works relevant to the topic in this section; see e.g. [117], [34], [45], [74], [113], [144], [76].

Laguerre-Galerkin method

Let $\omega_\alpha = x^\alpha e^{-x}$ and $\hat{\omega}_\alpha = x^\alpha$. We begin by considering the $L^2_{\omega_\alpha}$-orthogonal projection: $\pi_{N,\alpha} : L^2_{\omega_\alpha}(\mathbb{R}_+) \to P_N$, defined by

$$(u - \pi_{N,\alpha} u, v_N)_{\omega_\alpha} = 0, \quad \forall v_N \in P_N. \tag{4.4.1}$$

We derive from the orthogonality of $\mathcal{L}_n^{(\alpha)}$ (4.2.2) that

$$\pi_{N,\alpha} u = \sum_{n=0}^{N} \hat{u}_n^{(\alpha)} \mathcal{L}_n^{(\alpha)}, \quad \text{with } \hat{u}_n^{(\alpha)} = (u, \mathcal{L}_n^{(\alpha)})_{\omega_\alpha} / \gamma_n^{(\alpha)}.$$

Similar to the Jacobi approximations, we introduce

$$H^m_{\omega_\alpha,*}(\mathbb{R}_+) := \{u \,:\, \partial_x^k u \in L^2_{\omega_{\alpha+k}}(\mathbb{R}_+), \;\; 0 \leqslant k \leqslant m\}, \tag{4.4.2}$$

equipped with the norm and semi-norm

$$\|u\|_{H^m_{\omega_\alpha,*}} = \Big(\sum_{k=0}^{m} \|x^{k/2} \partial_x^k u\|^2_{\omega_\alpha} \Big)^{1/2}, \quad |u|_{H^m_{\omega_\alpha,*}} = \|x^{m/2} \partial_x^m u\|^2_{\omega_\alpha}.$$

Before presenting the main result, we make the observation that

$$\partial_x^k \mathcal{L}_n^{(\alpha)}(x) = (-1)^k \mathcal{L}_{n-k}^{(\alpha+k)}(x), \quad n \geqslant k, \tag{4.4.3}$$

which follows by using the derivative relation (4.2.6a) repeatedly. Hence, $\{\partial_x^k \mathcal{L}_n^{(\alpha)}\}$ are mutually orthogonal in $L^2_{\omega_{\alpha+k}}(\mathbb{R}_+)$, i.e.,

$$\int_0^{+\infty} \partial_x^k \mathcal{L}_n^{(\alpha)} \partial_x^k \mathcal{L}_m^{(\alpha)} \omega_{\alpha+k} \mathrm{d}x = \gamma_{n-k}^{(\alpha+k)} \delta_{mn}.$$

By Stirling's formula (1.8.12) and (4.2.3),

$$\gamma_{n-k}^{(\alpha+k)} = \frac{\Gamma(n+\alpha+1)}{\Gamma(n-k+1)} \sim n^{\alpha+k}, \quad \text{for } \; n \gg 1.$$

Then, using an argument similar to that in the proof of Theorem 1.8.1 leads to the fundamental approximation result for Laguerre polynomials:

Theorem 4.4.1 *For any $u \in H^m_{\omega_\alpha,*}(\mathbb{R}_+)$ and $m \geqslant 0$,*

$$\|\partial_x^l(\pi_{N,\alpha}u - u)\|_{\omega_{\alpha+l}} \lesssim N^{(l-m)/2}\|\partial_x^m u\|_{\omega_{\alpha+m}}, \quad 0 \leqslant l \leqslant m. \tag{4.4.4}$$

We now consider the corresponding approximation result for Laguerre functions. For any $u \in L^2_{\hat{\omega}_\alpha}(\mathbb{R}_+)$, we have $ue^{x/2} \in L^2_{\omega_\alpha}(\mathbb{R}_+)$. Let us denote

$$\widehat{P}_N = \text{span}\{\widehat{\mathcal{L}}_0^{(\alpha)}, \widehat{\mathcal{L}}_1^{(\alpha)}, \cdots, \widehat{\mathcal{L}}_N^{(\alpha)}\}, \tag{4.4.5}$$

and define the operator

$$\hat{\pi}_{N,\alpha}u = e^{-x/2}\pi_{N,\alpha}(ue^{x/2}) \in \widehat{P}_N, \tag{4.4.6}$$

where $\widehat{P}_N$ is given in (4.2.20). Clearly, by (4.4.1),

$$\begin{aligned}(\hat{\pi}_{N,\alpha}u - u, v_N)_{\hat{\omega}_\alpha} &= (\pi_{N,\alpha}(ue^{x/2}) - (ue^{x/2}), (v_N e^{x/2}))_{\omega_\alpha} \\ &= 0, \quad \forall v_N \in \widehat{P}_N.\end{aligned} \tag{4.4.7}$$

Hence, $\hat{\pi}_{N,\alpha}$ is the orthogonal projector from $L^2_{\hat{\omega}_\alpha}(\mathbb{R}_+)$ onto $\widehat{P}_N$.

Theorem 4.4.2 *Let $\hat{\partial}_x = \partial_x + \frac{1}{2}$. Then*

$$\|\hat{\partial}_x^l(\pi_{N,\alpha}u - u)\|_{\hat{\omega}_{\alpha+l}} \lesssim N^{(l-m)/2}\|\hat{\partial}_x^m u\|_{\hat{\omega}_{\alpha+m}}, \quad 0 \leqslant l \leqslant m. \tag{4.4.8}$$

Proof Let $v = ue^{x/2}$. It is clear that

$$\partial_x^l(\pi_{N,\alpha}v - v) = \partial_x^l(e^{x/2}(\hat{\pi}_{N,\alpha}u - u)) = e^{x/2}\hat{\partial}_x^l(\hat{\pi}_{N,\alpha}u - u),$$

and likewise, $\partial_x^m v = e^{x/2}\hat{\partial}_x^m u$. Hence, the desired result is a direct consequence of (4.4.4). □

Hereafter, let $\omega = e^{-x}$, and denote

$$H^1_{0,\omega}(\mathbb{R}_+) = \{u \in H^1_\omega(\mathbb{R}_+) \,:\, u(0) = 0\}, \quad P_N^0 = H^1_{0,\omega}(\mathbb{R}_+) \cap P_N.$$

Before we study the errors of the Laguerre-Galerkin approximation for (4.2.27), we need to establish a few lemmas. We shall first establish a Sobolev inequality and a Poincaré inequality in the semi-infinite interval.

Lemma 4.4.1 *For any given $v \in H^1_{0,\omega}(\mathbb{R}_+)$, we have*

$$\|e^{-\frac{x}{2}}v\|_{L^\infty(\mathbb{R}_+)} \leqslant \sqrt{2}\|v\|^{\frac{1}{2}}_{\omega}|v|^{\frac{1}{2}}_{1,\omega}, \qquad \|v\|_\omega \leqslant 2|v|_{1,\omega}. \tag{4.4.9}$$

Proof For any $x \in \mathbb{R}_+$,

$$\begin{aligned} e^{-x}v^2(x) &= \int_0^x \frac{\mathrm{d}}{\mathrm{d}y}(e^{-y}v^2(y))\mathrm{d}y \\ &= 2\int_0^x e^{-y}v(y)\frac{\mathrm{d}v(y)}{\mathrm{d}y}\mathrm{d}y - \int_0^x e^{-y}v^2(y)\mathrm{d}y, \end{aligned}$$

from which we derive

$$\begin{aligned} &e^{-x}v^2(x) + \int_0^x e^{-y}v^2(y)\mathrm{d}y \\ &\leqslant 2\int_0^\infty e^{-y}|v(y)\frac{\mathrm{d}v(y)}{\mathrm{d}y}|\mathrm{d}y \leqslant 2\|v\|_\omega|v|_{1,\omega}. \end{aligned} \tag{4.4.10}$$

This implies the first conclusion. Letting $x \to \infty$, we get the second inequality. □

Consider the orthogonal projection $\pi^{1,0}_N : H^1_{0,\omega}(\mathbb{R}_+) \to P^0_N$, defined by

$$((u - \pi^{1,0}_N u)', v'_N)_\omega = 0, \quad \forall v_N \in P^0_N. \tag{4.4.11}$$

Lemma 4.4.2 *If $u \in H^1_{0,\omega}(\mathbb{R}_+)$, and $\partial_x u \in H^{m-1}_{\omega,*}(\mathbb{R}_+)$, then for $m \geqslant 1$,*

$$\|\pi^{1,0}_N u - u\|_{1,\omega} \lesssim N^{\frac{1}{2}-\frac{m}{2}}\|x^{\frac{m-1}{2}}\partial^m_x u\|_\omega. \tag{4.4.12}$$

Proof Let $u_N(x) = \int_0^x \pi_{N-1,0}u'(y)\mathrm{d}y$. Then $u - u_N \in H^1_{0,\omega}(\mathbb{R}_+)$. It follows from Lemma 4.4.1 and Theorem 4.4.1 with $\alpha = 0$ that

$$\begin{aligned} \|\pi^{1,0}_N u - u\|_{1,\omega} &\leqslant \|u_N - u\|_{1,\omega} \lesssim |u - u_N|_{1,\omega} \\ &\lesssim N^{\frac{1}{2}-\frac{m}{2}}\|x^{\frac{m-1}{2}}\partial^m_x u\|_\omega. \end{aligned}$$

This ends the proof. □

Note that for any $u \in H^1_0(\mathbb{R}_+)$ we have $ue^{x/2} \in H^1_{0,\omega}(\mathbb{R}_+)$. Define the operator

$$\hat{\pi}^{1,0}_N u = e^{-x/2}\pi^{1,0}_N(ue^{x/2}) \in \widehat{P}^0_N, \quad \forall u \in H^1_0(\mathbb{R}_+).$$

The following lemma characterizes this operator.

Lemma 4.4.3 *For any $u \in H_0^1(\mathbb{R}_+)$, we have*

$$((u - \hat{\pi}_N^{1,0} u)', v_N') + \frac{1}{4}(u - \hat{\pi}_N^{1,0} u, v_N) = 0, \quad \forall v_N \in \widehat{P}_N^0. \tag{4.4.13}$$

Let $\hat{\partial}_x = \partial_x + \frac{1}{2}$. If $u \in H_0^1(\mathbb{R}_+)$ and $\hat{\partial}_x^m u \in L^2_{\hat{\omega}_{m-1}}(\mathbb{R}_+)$, then we have

$$\|\hat{\pi}_N^{1,0} u - u\|_1 \lesssim N^{\frac{1}{2}-\frac{m}{2}} \|\hat{\partial}_x^m u\|_{\hat{\omega}_{m-1}}. \tag{4.4.14}$$

Proof Using the definition of $\pi_N^{1,0}$, and integration by parts, we find that for any $v_N = w_N e^{-x/2}$ with $w_N \in P_N^0$,

$$\begin{aligned}
&((u - \hat{\pi}_N^{1,0} u)', v_N') \\
&= \Big([(ue^{x/2}) - \pi_N^{1,0}(ue^{x/2})]' - \frac{1}{2}[(ue^{x/2}) - \pi_N^{1,0}(ue^{x/2})], w_N' - \frac{1}{2} w_N \Big)_\omega \\
&= -\frac{1}{2} \int_0^\infty [(ue^{x/2} - \pi_N^{1,0}(ue^{x/2})) w_N]' e^{-x} \mathrm{d}x + \frac{1}{4}((ue^{x/2}) - \pi_N^{1,0}(ue^{x/2}), w_N)_\omega \\
&= -\frac{1}{4}((ue^{x/2}) - \pi_N^{1,0}(ue^{x/2}), w_N)_\omega = -\frac{1}{4}(u - \hat{\pi}_N^{1,0} u, v_N),
\end{aligned}$$

which implies the identity (4.4.13). Let $v = ue^{x/2}$. It is clear that

$$\partial_x(\hat{\pi}_N^{1,0} u - u) = -\frac{1}{2} e^{-x/2}(\pi_N^{1,0} v - v) + e^{-x/2} \partial_x(\pi_N^{1,0} v - v).$$

Hence, using Lemma (4.4.2) and the fact $\partial_x^m v = e^{x/2} \hat{\partial}_x^m u$, leads to

$$\begin{aligned}
\|\partial_x(\hat{\pi}_N^{1,0} u - u)\| &\lesssim \|\pi_N^{1,0} v - v\|_\omega + \|\partial_x(\pi_N^{1,0} v - v)\|_\omega \\
&\lesssim N^{\frac{1}{2}-\frac{m}{2}} \|x^{\frac{m-1}{2}} \partial_x^m v\|_\omega \lesssim N^{\frac{1}{2}-\frac{m}{2}} \|x^{\frac{m-1}{2}} \hat{\partial}_x^m u\|.
\end{aligned}$$

Similarly, we have

$$\|\hat{\pi}_N^{1,0} u - u\| \lesssim N^{\frac{1}{2}-\frac{m}{2}} \|x^{\frac{m-1}{2}} \hat{\partial}_x^m u\|.$$

This completes the proof. □

A complete error analysis for (4.2.31) needs error estimates for the laguerre-Gauss-Radau interpolation which are much more involved than the interpolation errors on a finite interval. We shall refer to [121] (and the references therein) where some optimal Laguerre interpolation error estimates were established. To simplify the presentation, we shall ignore the interpolation error here. We are now in a position to perform the error analysis.

Theorem 4.4.3 *Let $\hat{\partial}_x = \partial_x + \frac{1}{2}$, $\gamma > 0$, and let u and u_N be respectively the solution of* (4.2.27) *and* (4.2.31) *where $\hat{I}_N f$ is replaced f. Then, if $u \in H_0^1(\mathbb{R}_+)$ and $\hat{\partial}_x^m u \in L^2_{\hat{\omega}_{m-1}}(\mathbb{R}_+)$, we have*

$$\|u - u_N\|_1 \lesssim N^{\frac{1}{2}-\frac{m}{2}} \|\hat{\partial}_x^m u\|_{\hat{\omega}_{m-1}}. \tag{4.4.15}$$

Proof Let $e_N = u_N - \hat{\pi}_N^{1,0} u$ and $\tilde{e}_N = u - \hat{\pi}_N^{1,0} u$. Hence, by (4.2.30) and (4.2.31),

$$a(u_N - u, v_N) = 0, \quad \forall v_N \in \widehat{P}_N^0.$$

Due to (4.4.13), we have $a(e_N, v_N) = a(\tilde{e}_N, v_N) = (\gamma - 1/4)(\tilde{e}_N, v_N)$. Using the Cauchy-Schwarz inequality and Lemma 4.4.3 yields, for $\gamma > 0$,

$$\|e_N\|_1 \lesssim \|\tilde{e}_N\|_1 \lesssim N^{\frac{1}{2}-\frac{m}{2}} \|x^{\frac{m-1}{2}} \hat{\partial}_x^m u\|.$$

Finally, the estimate (4.4.15) follows from the triangle inequality and Lemma 4.4.3. □

Hermite-Galerkin method

The analysis for the Hermite case can be carried out in a similar fashion. Let $\omega = e^{-x^2}$ be the Hermite weight. We define the L^2_ω-orthogonal projection $\pi_N : L^2_\omega(\mathbb{R}) \to P_N$ by

$$(u - \pi_N u, v_N)_\omega = 0, \quad \forall v_N \in P_N.$$

We derive immediately from (4.1.2) that

$$\pi_N u(x) = \sum_{n=0}^{N} \hat{u}_n H_n(x),$$

with

$$\hat{u}_n = \frac{1}{\sqrt{\pi} 2^n n!} \int_{-\infty}^{\infty} u(x) H_n(x) e^{-x^2} \mathrm{d}x, \quad n \geqslant 0.$$

To obtain the approximation result for π_N, we observe from (4.1.4a) that

$$\partial_x^k H_n(x) = 2^k n(n-1)\cdots(n-k+1) H_{n-k}(x), \quad n \geqslant k, \tag{4.4.16}$$

which implies that $\{H_n\}$ are orthogonal under the inner product of the Sobolev space $H^m_\omega(\mathbb{R})$. Hence, using an argument similar to that for the proof of Theorem 1.8.1, we can easily establish the following result:

Theorem 4.4.4 *For any $u \in H^m_\omega(\mathbb{R})$ with $m \geqslant 0$,*

$$\|\partial_x^l(\pi_N u - u)\|_\omega \lesssim N^{(l-m)/2}\|\partial_x^m u\|_\omega, \quad 0 \leqslant l \leqslant m. \tag{4.4.17}$$

We now consider the corresponding approximation result for Hermite functions. Since $ue^{x^2/2} \in L^2_\omega(\mathbb{R})$ for any $u \in L^2(\mathbb{R})$. We define

$$\hat{\pi}_N u := e^{-x^2/2}\pi_N(ue^{x^2/2}) \in \hat{P}_N, \tag{4.4.18}$$

where $\hat{P}_N$ is given in (4.1.20). It is easy to check that $\hat{\pi}_N u$ is the $L^2(\mathbb{R})-$orthogonal projection of u onto $\hat{P}_N$ since

$$(u - \hat{\pi}_N u, v_N) = \big(ue^{x^2/2} - \hat{\pi}_N(ue^{x^2/2}), v_N e^{x^2/2}\big)_\omega = 0, \quad \forall v_N \in \hat{P}_N. \tag{4.4.19}$$

The following result is a direct consequence of Theorem 4.4.4.

Corollary 4.4.1 *Let $\hat{\partial}_x = \partial_x + x$. For any $\hat{\partial}_x^m u \in L^2(\mathbb{R})$ with $m \geqslant 0$,*

$$\|\hat{\partial}_x^l(\hat{\pi}_N u - u)\| \lesssim N^{(l-m)/2}\|\hat{\partial}_x^m u\|, \quad 0 \leqslant l \leqslant m. \tag{4.4.20}$$

With the help of the above approximation results, it is straightforward to establish error estimates for the Hermite-Galerkin approximation to Poisson type equations.

Exercise 4.4

Problem 1 Given $\alpha_1 > 0$ and $\alpha_2 > 0$. Let u and u_N be respectively the solutions of (4.2.43) and (4.2.44). Show that for $u \in H^2_0(\mathbb{R}_+)$ and $\partial_x^2 u \in H^m_{\hat{\omega}^{m-2}}(\mathbb{R}_+)$, with $m \geqslant 2$, we have

$$\|u - u_N\|_2 \lesssim N^{1-\frac{m}{2}}\|u\|_{m,\hat{\omega}^{m-2}}.$$

Chapter 5

Some applications in one space dimension

Contents

5.1 Pseudospectral methods for boundary layer problems 184
5.2 Pseudospectral methods for Fredholm integral equations . . 190
5.3 Chebyshev spectral methods for parabolic equations 196
5.4 Fourier spectral methods for the KdV equation 204
5.5 Fourier method and filters 214
5.6 Essentially non-oscillatory spectral schemes 222

In this chapter, we present applications of the spectral method to some typical one-dimensional problems. The first section is concerned with two-point boundary value problems with boundary layers. Spectral collocation methods have some advantages for handling this class of problems, but special tricks are needed to deal with extremely thin layers. The second section is concerned with the Fredholm integral equations. It will be demonstrated that the spectral method is almost as efficient as the standard product integration methods while producinf much more accurate approximations. In Section 5.3, we present a Chebyshev spectral method for parabolic equations, and in Section 5.4 we consider Fourier spectral methods for the KdV equation. In the final two sections, we discuss Fourier approximation to discontinuous functions, the use of the spectral filters, and the applications to nonlinear hyperbolic

conservation laws. The last section — essentially non-oscillatory spectral methods — requires some background in hyperbolic conservation laws. A good reference on this topic is the book by LeVeque[101].

5.1 Pseudospectral methods for boundary layer problems

A direct application of PS methods
Boundary layer resolving spectral methods
Transformed coefficients
Numerical experiments

The case when $\epsilon \ll 1$ in (2.4.1) is particularly interesting and challenging. Many different phenomena can arise in such problems, including boundary layers and complicated internal transition regions. The last few decades have witnessed substantial progress in the development of numerical methods for the solution of such problems and several packages, such as COLSYS[5], PASVAR [100], and MUS[122] are presently available.

In this section, we consider the case where thin boundaries are formed when $\epsilon \ll 1$. It is well-known that spectral methods are attractive in solving this type of problems thanks to the fact that the spectral collocation points are clustered at the boundary, more precisely, we have

$$|x_1 - x_0| = |x_N - x_{N-1}| = |\cos(\pi/N) - 1| \approx \frac{1}{2}\left(\frac{\pi}{N}\right)^2 \approx \frac{5}{N^2}\,.$$

In other words, the spacing between the collocation points near the boundaries is of order $\mathcal{O}(N^{-2})$, in contrast with $\mathcal{O}(N^{-1})$ spacing for finite differences or finite elements.

Although spectral methods are much more efficient than finite differences and finite elements in solving boundary layers, still a large N is required to obtain accurate solutions when ϵ is sufficiently small. In the past few years, several modified spectral methods have been proposed that are designed to resolve thin boundary layers,see e.g. [47], [64], [160].

A direct application of PS methods

We first use the `CODE PSBVP.2` in Section 2.4 to compute the numerical solution of the following problem with small parameter ϵ.

Example 5.1.1 *The following example has variable coefficients and the solution*

develops two boundary layers of width $\mathcal{O}(\epsilon)$ near the boundaries. The equation is

$$\epsilon u''(x) - xu'(x) - u = f(x),$$

where $f(x) = ((x+1)/\epsilon - 1)\exp(-(x+1)/\epsilon) - 2((x-1)/\epsilon + 1)\exp((x-1)/\epsilon)$. *The boundary conditions are* $u(-1) = 1$, $u(+1) = 2$.

It can be verified that the function $u(x) = e^{-(x+1)/\epsilon} + 2e^{(x-1)/\epsilon}$ satisfies the above ODE. It also satisfies the boundary conditions to machine precision (which is about 16-digits in double precision) for all values of $\epsilon \leqslant 0.05$. The following table contains the maximum errors for $\epsilon = 10^{-2}, 10^{-3}$ and 10^{-4}:

N	ϵ=10^{-2}	ϵ=10^{-3}	ϵ=10^{-4}
32	1.397e-03	4.388e+00	6.450e+01
64	1.345e-10	3.024e-01	2.792e+01
128	8.843e-14	1.598e-04	7.006e+00
256	5.372e-13	9.661e-13①	1.321e-01

We observe that the Chebyshev pseudospectral method fails to resolve the solution satisfactorily for $\epsilon = 10^{-4}$, even with $N = 256$.

For comparison, we solve the problem by using the central-difference finite difference method with $\epsilon = 10^{-2}$ and 10^{-3}, respectively. The following results show that even with 1000 grid points the boundary layers are not well resolved:

N	ϵ=10^{-2}	ϵ=10^{-3}	N	ϵ=10^{-2}	ϵ=10^{-3}
32	1.900e+00	1.879e+00	256	1.077e+00	1.987e+00
64	1.866e+00	1.941e+00	512	6.443e-01	1.954e+00
128	1.560e+00	1.972e+00	1024	3.542e-01	1.714e+00

Boundary layer resolving spectral methods

In order to resolve very thin boundary layers by using a reasonable number of unknowns N, we transform the singularly perturbed linear BVP (2.4.1) via the variable transformation $x \mapsto y(x)$ (or $x = x(y)$) into the new BVP

$$\epsilon v''(y) + P(y)v'(y) + Q(y)v(y) = F(y), \tag{5.1.1}$$

where v is the transplant of u, $v(y) = u(x(y))$. The transformed coefficients are

$$P(y) = \frac{p(x)}{y'(x)} + \epsilon\frac{y''(x)}{y'(x)^2}, \quad Q(y) = \frac{q(x)}{y'(x)^2}, \quad F(y) = \frac{f(x)}{y'(x)^2}, \tag{5.1.2}$$

① We used `CODE DM.3` in Section 2.1 to compute the differentiation matrix D^1. That is, we have used the formulas (2.1.15) and (2.1.16). If we use more accurate formulas (2.1.17), this error will reduce to 6.84e-14.

where again $x = x(y)$. It is clear from (5.1.1) and (5.1.2) that for any variable transformation $x \mapsto y(x)$ the two quantities $1/y'(x)$ and $y''(x)/[y'(x)]^2$ are of interest and should be easy to calculate.

We now introduce the iterated sine functions $x = g_m(y), m = 0, 1, \cdots$, where

$$g_0(y) := y, \qquad g_m(y) = \sin\left(\frac{\pi}{2} g_{m-1}(y)\right), \quad m \geqslant 1. \tag{5.1.3}$$

The following result characterizes these transformations based on the relative spacing of the transformed Chebyshev points.

The following two statements hold for any integer $m \geqslant 0$:

(a) The map g_m is one–to–one and $g_m([-1, 1]) = [-1, 1]$;

(b) If the x_j are Chebyshev points $x_j = \cos(\pi j/N)$, then

$$\begin{aligned} g_m(x_0) - g_m(x_1) &= g_m(x_{N-1}) - g_m(x_N) \\ &= \frac{8}{\pi^2}\left(\frac{\pi^2}{4N}\right)^{2^{m+1}} \left(1 + \mathcal{O}(N^{-2})\right). \end{aligned} \tag{5.1.4}$$

For part (a) we need to show that $g_m'(y) \neq 0$ for $y \in (-1, 1)$, $|g_m(y)| \leqslant 1$ and $g_m(\pm 1) = \pm 1$, which can be proved by mathematical induction. Part (b) can also be established by induction (with respect to m).

Transformed coefficients

We now consider the transformation $x = x(y) := g_m(y)$. From (5.1.4) it can be expected that the transformations (5.1.3) together with the Chebyshev PS method can deal with extremely small boundary layers using a fairly small number of collocation points. For $m = 1, 2$ and 3 (which correspond to one, two and three sine transformations), the distance between each boundary point and its nearest interior point is $\mathcal{O}(N^{-4}), \mathcal{O}(N^{-8})$ and $\mathcal{O}(N^{-16})$, respectively. Therefore, even for very small ϵ such as $\epsilon = 10^{-12}$, at least one collocation point lies in the boundary layer even for moderate values of N, if two or three sine transformations are used.

After having the transformation $x(y) = g_m(y)$, we need to work on the transformed coefficients $P(y), Q(y)$ and $F(y)$ given by (5.1.2). The computation of $1/y'(x)$ is straightforward. Differentiating the recursion (5.1.3) we obtain

$$g_0'(y) = 1, \quad g_m'(y) = \frac{\pi}{2}\cos\left(\frac{\pi}{2} g_{m-1}(y)\right) g_{m-1}'(y), \quad m \geqslant 1. \tag{5.1.5}$$

Since $y'(x) = 1/g'_m(y)$, we have

$$\frac{1}{y'(x)} = \prod_{k=0}^{m-1} \left(\frac{\pi}{2} \cos\left(\frac{\pi}{2} g_k(y) \right) \right), \quad m \geqslant 1. \tag{5.1.6}$$

Further we define the functions $h_m(x)$, mapping $[-1, 1]$ onto itself, recursively via

$$h_0(x) := x, \qquad h_m(x) := \frac{2}{\pi} \arcsin\left(h_{m-1}(x) \right), \quad m \geqslant 1. \tag{5.1.7}$$

We will show that $h_m = g_m^{-1}$, for $m = 0, 1, \cdots$ (this implies that $y(x) = h_m(x)$). The case $m = 0$ is trivial. For $m \geqslant 1$, we let $z = h_m(g_m(y))$. It can be shown by induction that

$$g_k(z) = h_{m-k}(g_m(y)), \qquad 0 \leqslant k \leqslant m. \tag{5.1.8}$$

For $k = m$ we therefore obtain

$$g_m(z) = h_0(g_m(y)) = g_m(y), \tag{5.1.9}$$

and, since g_m is injective, it follows that $y = z$, i.e. $y = h_m(g_m(y))$.

We now proceed to find a recursion for the quantity $h''_m(x)/[h'_m(x)]^2$. From (5.1.7) we obtain

$$\sin\left(\frac{\pi}{2} h_m(x) \right) = h_{m-1}(x), \qquad m \geqslant 1. \tag{5.1.10}$$

Differentiating the above equation twice with respect to x yields

$$\begin{aligned} &\frac{\pi}{2} \cos\left(\frac{\pi}{2} h_m(x) \right) h'_m(x) = h'_{m-1}(x), \\ &-\left(\frac{\pi}{2} \right)^2 \sin\left(\frac{\pi}{2} h_m(x) \right) \left(h'_m(x) \right)^2 + \left(\frac{\pi}{2} \right) \cos\left(\frac{\pi}{2} h_m(x) \right) h''_m(x) = h''_{m-1}(x). \end{aligned} \tag{5.1.11}$$

Finally, using the above results we obtain the recursion

$$\frac{h''_m(x)}{\left(h'_m(x) \right)^2} = \frac{\pi}{2} \tan\left(\frac{\pi}{2} h_m(x) \right) + \frac{\pi}{2} \cos\left(\frac{\pi}{2} h_m(x) \right) \frac{h''_{m-1}(x)}{\left(h'_{m-1}(x) \right)^2}. \tag{5.1.12}$$

Note that $h'_0(x) \equiv 1$ and $h''_0(x) \equiv 0$. Since $y(x) = h_m(x)$, the quantity $y''(x)/[y'(x)]^2$ can be computed easily using (5.1.12).

Using (5.1.6) and (5.1.12), we are able to compute the coefficients $P(y), Q(y)$ and $F(y)$ in the transformed equation (5.1.1). The pseudocode for solving (5.1.1) is provided below:

```
CODE Layer.1
Input M, N, ε, p(x), q(x), f(x), β_L, β_R
Collocation points:  x(j)=cos(πj/N)
%first order differentiation matrix
call CODE DM.3 in Section 2.1 to get D1
%compute second order differentiation matrix
D2=D1*D1
% compute x=g_m(y) and 1/y'(x) at grid points
for j=1 to N-1 do
  gm=y(j);  yp(j)=1
    for mm=1 to M do
      yp(j)=(π/2)*cos(π*gm/2)
      gm=sin(π*gm/2)
    endfor
  x(j)=gm
% compute h_m(x) and y''(x)/[y'(x)]^2
   hm=x(j); hd(j)=0;
     for mm=1 to M do
       hm=(2/π)*asin(hm)
       hd(j)=(π/2)*tan(π*hm/2)+(π/2)*cos(π*hm/2)*hd(j)
     endfor
endfor
% compute the stiffness matrix A
for i=1 to N-1 do
   P(i)=p(x(i))*yp(i)+ε*hd(i)
   Q(i)=q(x(i))*(yp(i))^2; F(i)=f(x(i))*(yp(i))^2
      ss1=ε*D2(i,0)+P(i)*D1(i,0);
      ss2=ε*D2(i,N)+P(i)*D1(i,N);
         for j=1 to N-1 do
             if i=j
                A(i,j)=ε*D2(i,j)+P(i)*D1(i,j)+Q(i)
             else
                A(i,j)=ε*D2(i,j)+P(i)*D1(i,j)
             endif
         endfor
% compute the right-hand side vector b
   b(i)=F(i)-ss1*β_R-ss2*β_L
endfor
% solve the linear system to get the unknown vector
```

```
u=A^{-1}b
Output u(1), u(2), ···, u(N-1)
```

The MATLAB code based on the above algorithm is given below.

```
CODE Layer.2
Input eps, M, N, p(x), q(x), f(x), betaL, betaR
pi1=pi/2;
j=[1:1:N-1];  y=[cos(pi*j/N)]';
% MATLAB code for DM1 is given by CODE DM.4 in Section 2.1
D1=DM1(N);  D2=D1^2;
for j=1:N-1
   gm=y(j);  yp(j)=1;
      for mm=1:M
         yp(j)=yp(j)*pi1*cos(pi1*gm); gm=sin(pi1*gm);
      end
    x(j)=gm;
%compute y''(x)/[y'(x)]^2
   hm=x(j);  hd(j)=0;
      for mm=1:M
         hm=asin(hm)/pi1;
         hd(j)=pi1*tan(pi1*hm)+pi1*cos(pi1*hm)*hd(j);
      end
end
% compute the stiffness matrix
for i=1:N-1
   P1=p(x(i))*yp(i)+eps*hd(i);
   Q1=q(x(i))*yp(i)^2;  F1=f(x(i))*yp(i)^2;
      for j=1:N-1
        if i==j
          A(i,j)=eps*D2(i+1,j+1)+P1*D1(i+1,j+1)+Q1;
        else
          A(i,j)=eps*D2(i+1,j+1)+P1*D1(i+1,j+1);
        end if
      end for
   ss1=eps*D2(i+1,1)+P1*D1(i+1,1);
   ss2=eps*D2(i+1,N+1)+P1*D1(i+1,N+1);
   b(i)=F1-ss1*betaR-ss2*betaL;
end
% solve the linear system
u=A\b';
```

Numerical experiments

We compute the numerical solutions for Example 5.1.1 using the above MATLAB code. The following table contains the results of our experiments for $\epsilon = 10^{-3}, \epsilon = 10^{-6}$ and $\epsilon = 10^{-9}$. We use two sine iterations in the above code.

N	ϵ=10^{-3}	ϵ=10^{-6}	ϵ=10^{-9}
32	1.96e-02	4.77e+00	6.56e-01
64	3.91e-04	2.11e-01	3.20e-01
128	1.74e-09	7.48e-03	3.03e-01
256	8.66e-12	6.82e-07	9.00e-02
512	1.25e-09	3.87e-10	6.24e-05

The maximum errors in the above table suggest that with two sine iterations very thin boundary can be resolved with $\mathcal{O}(100)$ collocation points.

We close by pointing out that there have been various types of methods for handling singular solutions or coordinate singularities with spectral methods see, e.g. [136], [42], [85], [48], [103], [77], [104].

Exercise 5.1

Problem 1 Solve the boundary value problem in Problem 1 of Exercise 2.4 with `CODE Layer.1`, and compare the errors thus obtained.

5.2 Pseudospectral methods for Fredholm integral equations

Trapezoidal method and simpson's method
Integral of Lagrange polynomials
Linear system
Computational efficiency

We consider the Fredholm integral equation of the second kind,

$$u(x) + \int_{-1}^{1} K(x,s)u(s)\mathrm{d}s = g(x), \qquad x \in [-1,1], \tag{5.2.1}$$

where the kernel functions $K(x,s)$ and $g(x)$ are given. There exist many product-integration type numerical methods for the solution of (5.2.1), such as second-order trapezoidal method, fourth-order Simpson's method, see e.g. Brunner[26], Davis[37] and Atkinson[6]. In this section, we describe a method based on the Chebyshev pseudospectral method. For comparison, we begin by introducing the trapezoidal and

Simpson's methods.

Trapezoidal method and Simpson's method

We divide $[-1,1]$ into N equal intervals by the points

$$-1 = y_0 < y_1 < \cdots < y_{N-1} < y_N = 1.$$

Specifically, if $h = 2/N$ is the common length of the intervals, then $y_j = -1 + jh$, $0 \leqslant j \leqslant N$. The so-called *composite trapezoidal rule* for the integral of a given function $f(s)$ is defined by

$$\int_{-1}^{1} f(s)\mathrm{d}s \cong h\left(\frac{f(y_0)}{2} + f(y_1) + \cdots + f(y_{N-1}) + \frac{f(y_N)}{2}\right). \tag{5.2.2}$$

Using the composite trapezoidal rule to treat the integral term in (5.2.1) leads to the following trapezoidal method:

$$U(y_j) + h\sum_{k=0}^{N} \tilde{c}_k^{-1} K(y_j, y_k)U(y_k) = g(y_j), \qquad 0 \leqslant j \leqslant N,$$

where $\tilde{c}_j = 1$ except $\tilde{c}_0 = \tilde{c}_N = 2$. Solving the above linear system will give an approximate solution for (5.2.1). The convergence rate for this approach is two.

Similarly, the *composite Simpson's rule* is given by

$$\begin{aligned}\int_{-1}^{1} f(s)\mathrm{d}s \cong \frac{h}{3}\Big(&f(y_0) + 4f(y_1) + 2f(y_2) + 4f(y_3) + 2f(y_4) + \cdots \\ &+2f(y_{N-2}) + 4f(y_{N-1}) + f(y_N)\Big),\end{aligned} \tag{5.2.3}$$

where N is an even[①], positive integer. Using the composite Simpson's rule to treat the integral term in (5.2.1) leads to the following Simpson's method:

$$U(y_j) + \frac{h}{3}\sum_{k=0}^{N} c_k K(y_j, y_k)U(y_k) = g(y_j), \qquad 0 \leqslant j \leqslant N.$$

Here $c_0 = 1, c_N = 1$, $c_k = 4$ for $1 \leqslant k \leqslant N-1$ and k odd, $c_k = 2$ for $1 \leqslant k \leqslant N-1$ and k even. The convergence rate for this approach is four.

To obtain a better understanding of the above two methods, they will be employed to solve the following example.

① For simplicity, it is always required that N, the number of sub-intervals, is even for Simpson's method.

Example 5.2.1 *Consider* (5.2.1) *with* $K(x,s) = e^{xs}$ *and*

$$g(x) = e^{4x} + \left(e^{x+4} - e^{-(x+4)}\right)/(x+4).$$

The exact solution is $u(x) = e^{4x}$.

The maximum errors obtained by using the trapezoidal and Simpson's methods are listed below:

N	Trapezoidal method	Simpson's method
8	7.878e-01	5.782e-02
16	3.259e-01	7.849e-03
32	1.022e-01	6.825e-04
64	2.846e-02	4.959e-05
128	7.500e-03	3.329e-06
256	1.924e-03	2.154e-07

The second-order convergence rate for the trapezoidal method and the fourth-order convergence rate for the Simpson's method are observed from the above table.

Integral of Lagrange polynomials

In our computation, we need the values of the integrals

$$b_k = \int_{-1}^{1} T_k(s)\mathrm{d}s, \qquad 0 \leqslant k \leqslant N.$$

They can be computed using (1.3.4),

$$\begin{aligned} &2b_k = \frac{1}{k+1}\Big(T_{k+1}(1) - T_{k+1}(-1)\Big) - \frac{1}{k-1}\Big(T_{k-1}(1) - T_{k-1}(-1)\Big), \quad k \geqslant 2, \\ &b_0 = 2, \quad b_1 = 0. \end{aligned} \tag{5.2.4}$$

Since $T_k(\pm 1) = (\pm 1)^k$, we obtain

$$b_k = \begin{cases} 0, & k \quad \text{odd}, \\ 2/(1-k^2), & k \quad \text{even}. \end{cases} \tag{5.2.5}$$

As before, the Chebyshev points are defined by $x_j = \cos(\pi j/N)$, $0 \leqslant j \leqslant N$. For a fixed k, $0 \leqslant k \leqslant N$, let $F_k(s)$ be the polynomial of minimal degree which takes the value 1 at $s = x_k$ and 0 at $s = x_j$, $j \neq k$ (i.e. $F_k(s)$ is the Lagrange polynomial). Expand $F_k(s)$ in terms of the Chebyshev polynomials, i.e., $F_k(s) =$

$\sum_{j=0}^{N} a_{kj} T_j(s), s \in [-1,1]$. Similar to the derivation of (5.3.4), we obtain

$$a_{kj} = \frac{2}{N\tilde{c}_j} \sum_{m=0}^{N} \tilde{c}_m^{-1} F_k(x_m) \cos(jm\pi/N) = \frac{2}{N\tilde{c}_j \tilde{c}_k} \cos(jk\pi/N).$$

The above result gives

$$F_k(s) = \frac{2}{N\tilde{c}_k} \sum_{j=0}^{N} \tilde{c}_j^{-1} \cos(jk\pi/N)\, T_j(s).$$

Integrating on both sides from -1 to 1 leads to

$$d_k := \int_{-1}^{1} F_k(s)\mathrm{d}s = \frac{2}{N\tilde{c}_k} \sum_{j \text{ even}} \tilde{c}_j^{-1} b_j \cos(jk\pi/N), \tag{5.2.6}$$

where b_j is defined by (5.2.5).

Linear system

We can now use the Chebyshev pseudospectral method to solve (5.2.1). The trick is that we expand the functions in the integrand, rather than expanding the unknown function only:

$$K(x,s)u(s) \approx \sum_{k=0}^{N} F_k(s) K(x,s_k) U(s_k), \quad s_k = \cos(\pi k/N), \quad 0 \leqslant k \leqslant N. \tag{5.2.7}$$

where $U(s_k)$ is the approximation for $u(s_k)$, $0 \leqslant k \leqslant N$. We then use (5.2.7) and let (5.2.1) hold at the Chebyshev points $\{x_j\}$:

$$U(x_j) + \sum_{k=0}^{N} K(x_j, s_k) U(s_k) \int_{-1}^{1} F_k(s)\mathrm{d}s = g(x_j), \qquad 0 \leqslant j \leqslant N.$$

We write the above equations into matrix form,

$$\begin{pmatrix} U(x_0) \\ \vdots \\ U(x_N) \end{pmatrix} + \mathcal{M}(x,s) \begin{pmatrix} U(s_0) \\ \vdots \\ U(s_N) \end{pmatrix} = \begin{pmatrix} g(x_0) \\ \vdots \\ g(x_N) \end{pmatrix}, \tag{5.2.8}$$

where the matrix $\mathcal{M}(x,s)$ is defined by

$$\mathcal{M}(x,s)=\begin{pmatrix} d_0K(x_0,s_0) & d_1K(x_0,s_1) & \cdots & d_NK(x_0,s_N)\\ d_0K(x_1,s_0) & d_1K(x_1,s_1) & \cdots & d_NK(x_1,s_N)\\ \vdots & \vdots & & \vdots\\ d_0K(x_N,s_0) & d_1K(x_N,s_1) & \cdots & d_NK(x_N,s_N)\end{pmatrix}, \tag{5.2.9}$$

with d_k given by (5.2.6). Since $s_k=x_k$, $0\leqslant k\leqslant N$, we obtain

$$(I+\mathcal{M}(x,x))\vec{U}=\vec{g}, \tag{5.2.10}$$

where $\vec{U}=(U(x_0),\cdots,U(x_N))^{\mathrm{T}}$ and $\vec{g}=(g(x_0),\cdots,g(x_N))^{\mathrm{T}}$.

A pseudocode is given below:

```
CODE Intgl.1
Input N, K(x,s), g(x)
Compute the collocation points:  x(k)=cos(πk/N)
   x(k)=cos(πk/N);  c̃(k)=1
end
c̃(0)=2;  c̃(N)=2; c̃(k)=1, 1⩽k⩽N-1
% compute the integral of the Chebyshev function, b_k
for k=0 to N do
   if k is even then  b(k)=2/(1-k^2)
   else b(k)=0
   endif
endfor
% compute d_k
for k=0 to N do
   dd=0
      for j=0 to N do
         dd=dd+b(j)*cos(jkπ/N)/c̃(j)
      endfor
    d(k)=2*dd/(N*c̃(k))
 endfor
 % form the stiffness matrix
 for i=0 to N do
    for j=0 to N do
       if i=j then  A(i,j)=1+d(i)*K(x(i),x(i))
       else  A(i,j)=d(j)*K(x(i),x(j))
       endif
    endfor
  % form the right-hand side vector
```

```
    b(i)=g(x(i))
 endfor
% solve the linear system
u=A^{-1}b
Output u(0), u(1), ···, u(N)
```

Using the above code to compute the numerical solutions of Example 5.2.1 gives the following results (the maximum errors):

N	Maximum error	N	Maximum error
6	1.257e-02	14	1.504e-09
8	2.580e-04	16	1.988e-11
10	5.289e-06	18	2.416e-13
12	9.654e-08	20	2.132e-14

Computational efficiency

For differential equations, numerical methods based on the assumption that the solution is approximately the low-order polynomials lead to a *sparse* system of algebraic equations. Efficient softwares are available to compute the solutions of the system. In contrast, a similar approach for the integral equation (5.2.1) leads to a full system. Even with the (low-order) trapezoidal method the stiffness matrix is full.

It is clear that the spectral method is much more accurate than the trapezoidal and Simpson's methods. As for computational efficiency, the only extra time used for the spectral method is the calculation of d_k (see (5.2.6)). By using FFT, it means that the extra cost is about $\mathcal{O}(N \log N)$ operations. In other words, the computational time for the three methods discussed above are almost the same.

In summary, the spectral method is almost as efficient as the standard product integration methods, but it produces much more accurate approximations.

Exercise 5.2

Problem 1 Assume that $f(\theta)$ is periodic in $[0, 2\pi]$. If f is smooth, it can be shown that the trapezoid rule (5.2.2) converges extremely fast (i.e. exponentially). The rapid convergence of the periodic trapezoid rule can be found in [81]. For illustration, evaluate

$$I = \int_0^{2\pi} \sqrt{\frac{1}{4}\sin^2\theta + \cos^2\theta}\mathrm{d}\theta.$$

How many terms have to be used to get the 12-digits correct ($I = 4.8442241102738\cdots$)?

Problem 2 Consider the following integro-differential equation:

$$x^2u'(x)+e^xu(x)+\int_{-1}^{1}e^{(x+1)s}u(s)\mathrm{d}s=f(x),\qquad -1\leqslant x\leqslant 1,$$
$$u(-1)+u(1)=e+e^{-1}.$$

Choose $f(x)=(x^2+e^x)e^x+(e^{x+2}-e^{-(x+2)})/(x+2)$ so that the exact solution is $u(x)=e^x$. Solve this problem by using the spectral techniques studied in this chapter, with $N=4,6,8,10,12,14$. Plot the maximum errors.

Problem 3 Consider the following integro-differential equation:

$$e^xu''(x)+\cos(x)u'(x)+\sin(x)u(x)+\int_{-1}^{1}e^{(x+1)s}u(s)\mathrm{d}s=g(x),\qquad -1\leqslant x\leqslant 1$$
$$u(1)+u(-1)+u'(1)=2e+e^{-1},\qquad u(1)+u(-1)-u'(-1)=e.$$

Choose $f(x)=(e^x+\cos(x)+\sin(x))e^x+(e^{x+2}-e^{-(x+2)})/(x+2)$ so that the exact solution is $u(x)=e^x$. Solve this problem by using the spectral techniques studied in this chapter, with $N=4,6,8,10,12,14$. Plot the maximum errors.

5.3 Chebyshev spectral methods for parabolic equations

Derivatives and their coefficients
Linear heat equation
Nonlinear Burgers' equation

Let us begin with the simple case, the linear heat equation with homogeneous boundary conditions:

$$u_t=u_{xx},\qquad x\in(-1,1);\qquad u(\pm1,t)=0. \tag{5.3.1}$$

An initial condition is also supplied. We can construct a Chebyshev method for the heat equation as follows:

`Step 1` Approximate the unknown function $u(x,t)$ by

$$u^N(x,t)=\sum_{k=0}^{N}a_k(t)T_k(x), \tag{5.3.2}$$

where $T_k(x)$ are the Chebyshev polynomials.

`Step 2` Let $\{x_j\}$ be the Chebyshev-Gauss-Lobatto points $x_j = \cos(\pi j/N)$, $0 \leqslant j \leqslant N$. Substituting the above polynomial expansion into the heat equation and assuming that the resulting equation holds at $\{x_j\}_{j=1}^{N-1}$, we find

$$\frac{du^N}{dt}(x_j,t) = \sum_{k=0}^{N} a_k(t) T_k''(x_j), \quad 1 \leqslant j \leqslant N-1; \quad u^N(\pm 1, t) = 0. \tag{5.3.3}$$

Notice that $\{a_k\}$ can be determined *explicitly* by $\{u^N(x_j,t)\}$ from (5.3.2), namely, we derive from (1.3.17) that

$$a_k(t) = \frac{2}{N\tilde{c}_k} \sum_{j=0}^{N} \tilde{c}_j^{-1} u^N(x_j,t) \cos(\pi jk/N), \qquad 0 \leqslant k \leqslant N, \tag{5.3.4}$$

where $\tilde{c}_0 = 2, \tilde{c}_N = 2$ and $\tilde{c}_j = 1$ for $1 \leqslant j \leqslant N-1$. Thus, combining the above two equations gives the following system of ODE:

$$\frac{d}{dt} u^N(x_j,t) = G_j\Big(u^N(x_0,t), u^N(x_1,t), \cdots, u^N(x_N,t)\Big), \qquad 1 \leqslant j \leqslant N-1,$$

where G_j can be expressed explicitly.

Derivatives and their coefficients

Notice that $\deg(u^N) \leqslant N$. The first derivative of u_N has the form

$$\frac{\partial u^N}{\partial x} = \sum_{k=0}^{N-1} a_k^{(1)}(t) T_k(x), \tag{5.3.5}$$

where the expansion coefficients $a_k^{(1)}$ will be determined by the coefficients a_k in (5.3.2). It follows from $T_0'(x) \equiv 0$ and (5.3.2) that

$$\frac{\partial u^N}{\partial x} = \sum_{k=1}^{N-1} a_k(t) T_k'(x). \tag{5.3.6}$$

On the other hand, (1.3.4) and (5.3.5) lead to

$$\frac{\partial u^N}{\partial x} = \sum_{k=0}^{N-1} a_k^{(1)}(t) T_k(x)$$

$$
\begin{aligned}
&= a_0^{(1)} T_1'(x) + \frac{1}{4} a_1^{(1)} T_2'(x) + \frac{1}{2} \sum_{k=2}^{N} a_k^{(1)}(t) \left[\frac{1}{k+1} T_{k+1}'(x) - \frac{1}{k-1} T_{k-1}'(x) \right] \\
&= a_0^{(1)} T_1'(x) + \sum_{k=2}^{N} \frac{1}{2k} a_{k-1}^{(1)} T_k'(x) - \sum_{k=1}^{N-1} \frac{1}{2k} a_{k+1}^{(1)} T_k'(x) \\
&= \sum_{k=1}^{N} \frac{1}{2k} \left(\tilde{c}_k a_{k-1}^{(1)} - a_{k+1}^{(1)} \right) T_k'(x),
\end{aligned}
\tag{5.3.7}
$$

where we have assumed that $a_N^{(1)} = a_{N+1}^{(1)} = 0$. By comparing (5.3.6) and (5.3.7), we obtain

$$
\tilde{c}_k a_k^{(1)}(t) = a_{k+2}^{(1)}(t) + 2(k+1) a_{k+1}(t), \quad k = N-1, N-2, \cdots, 0,
$$

where $a_{N+1}^{(1)}(t) \equiv 0, a_N^{(1)}(t) \equiv 0$. If we write the higher-order derivatives in the form

$$
\frac{\partial^m u^N}{\partial x^m} = \sum_{k=0}^{N-m} a_k^{(m)}(t) T_k(x), \quad m \geqslant 1,
$$

a similar procedure as above will give the following recurrence formula:

$$
\begin{aligned}
&\tilde{c}_k a_k^{(m)}(t) = a_{k+2}^{(m)}(t) + 2(k+1) a_{k+1}^{(m-1)}(t), \quad k = N-m, N-m-1, \cdots, 0, \\
&a_{N+1}^{(m)}(t) \equiv 0, \quad a_N^{(m)}(t) \equiv 0, \quad \text{for} \quad m \geqslant 1.
\end{aligned}
\tag{5.3.8}
$$

Linear heat equation

We first consider the heat equation (5.3.1) with the initial condition

$$
u(x,0) = u_0(x), \qquad x \in (-1,1). \tag{5.3.9}
$$

We solve the above problem by using the spectral method in space. For ease of illustration for the spectral method, we employ the forward Euler method in time direction. High-order accuracy temporal discretization will be discussed later. By use of (5.3.5), the model problem (5.3.1) becomes

$$
\frac{\partial u^N}{\partial t} \Big|_{x=x_j} = \sum_{k=0}^{N} a_k^{(2)}(t) \cos(\pi j k / N). \tag{5.3.10}
$$

The procedure for using the above formula involving two FFTs:

• Use FFT to evaluate $a_k(t_n)$, which will be used to evaluate $a_k^{(2)}(t_n)$ with small amount $\mathcal{O}(N)$ operations based on (5.3.8);

• Then FFT can be used again to evaluate the right-hand side of (5.3.10).

The following pseudocode implements the numerical procedure. The ODE system (5.3.10) is solved by the forward Euler method.

```
CODE Exp.1
Input N, u_0(x), Δt, Tmax
%collocation points, initial data, and c̃_k
for j=0 to N do
    x(j)=cos(πj/N), u(j)=u_0(x(j)), c̃(j)=1
endfor
c̃(0)=2, c̃(N)=2
%set starting time    time=0
While time ⩽ Tmax do
     % Need to call F(u), the RHS of the ODE system
     rhs=RHS(u,N,c̃)
     %Forward Euler method
     for j=1 to N-1 do
        u(j)=u(j)+Δt*RHS(j)
     endfor
   %set new time level
   time=time+Δt
endwhile
Output u(1),u(2),···,u(N-1)
```

The right-hand side of (5.3.10) is given by the following subroutine:

```
CODE Exp.2
function r=RHS(u,N,c̃)
%calculate coefficients a_k(t)
   for k=0 to N do
      a(0,k)=2/(N*c̃(k))Σ_{j=0}^{N} u(j)cos(πjk/N)/c̃(j)
   endfor
%calculate coefficients a_k^{(i)}(t), i=1, 2
   for i=1 to 2 do
     a(i,N+1)=0,  a(i,N)=0
     for k=N-1 to 0 do
       a(i,k)=(a(i,k+2)+2(k+1)*a(i-1,k+1))/c̃(k)
     endfor
   endfor
```

```
%calculate the RHS function of the ODE system
   for j=0 to N do
     r(j)=∑_{k=0}^{N} a(2,k)cos(πjk/N)
   endfor
```

Example 5.3.1 *Consider the model problem* (5.3.1) *with the initial function* $u_0(x) = \sin(\pi x)$. *The exact solution is given by* $u(x,t) = e^{-\pi^2 t}\sin(\pi x)$.

The following is the output with $T_{\max} = 0.5$:

N	$\|e\|_\infty(\Delta t=10^{-3})$	$\|e\|_\infty(\Delta t=10^{-4})$	N	$\|e\|_\infty(\Delta t=10^{-3})$	$\|e\|_\infty(\Delta t=10^{-4})$
3	1.083e-02	1.109e-02	7	1.635e-04	1.855e-05
4	3.821e-03	3.754e-03	8	1.642e-04	1.808e-05
5	7.126e-04	8.522e-04	9	1.741e-04	1.744e-05
6	2.140e-04	5.786e-05	10	1.675e-04	1.680e-05

It is observed that for fixed values of Δt the error decreases until $N = 7$ and then remains almost unchanged. This implies that for $N \geqslant 7$, the error is dominated by that of the time discretization. Due to the small values of the time step, the effect of rounding errors can also be observed.

For comparison, we also compute finite-difference solutions for the model problem (5.3.1). We use the equal space mesh $-1 = x_0 < x_1 < \cdots < x_N = 1$, with $x_j = x_{j-1} + 2/N, 1 \leqslant j \leqslant N$. The central-differencing is employed to approximate the spatial derivative u_{xx} and the forward Euler is used to approximate the time derivative u_t. It is well known that the error in this case is $\mathcal{O}(\Delta t + N^{-2})$. Below is the output:

N	$\|e\|_\infty(\Delta t=10^{-3})$	$\|e\|_\infty(\Delta t=10^{-4})$	N	$\|e\|_\infty(\Delta t=10^{-3})$	$\|e\|_\infty(\Delta t=10^{-4})$
3	2.307e-02	2.335e-02	10	1.007e-03	1.163e-03
5	5.591e-03	5.792e-03	15	3.512e-04	5.063e-04
7	2.464e-03	2.639e-03	20	1.185e-04	2.717e-04

We now make some observations from the above two tables. Assume that a highly accurate and stable temporal discretization is used. In order that the maximum error of the numerical solution to be of order $\mathcal{O}(10^{-5})$, the spectral method requires that $N = 7$ (i.e. 6 grid points inside the space direction), but the central finite-difference method needs more than 40 points. The difference in the number of grid points will become much larger if smaller error bound is used.

Nonlinear Burgers' equation

By slightly modifying the pseudocode (the boundary conditions and the right-hand side functions need to be changed), we can handle nonlinear non-homogeneous

problems. We demonstrate this by considering a simple example below.

In the area of computational fluid dynamics (CFD), the Burgers' equation

$$u_t + uu_x = \epsilon u_{xx} \tag{5.3.11}$$

is a popular model equation. It contains the nonlinear convection term uu_x and the diffusion term ϵu_{xx}. Let

$$u = 2\epsilon \frac{\partial}{\partial x}(\ln \psi).$$

Then (5.3.11) becomes the (linear) diffusion equation $\psi_t = \epsilon \psi_{xx}$, which allows an analytical solution. For example,

$$u(x,t) = \frac{\epsilon}{1+\epsilon t}\left[x + \tan\left(\frac{x}{2(1+\epsilon t)}\right)\right]. \tag{5.3.12}$$

Example 5.3.2 *Consider the Burgers' equation* (5.3.11) *on* $(-1,1)$ *with boundary conditions and initial condition such that the exact solution is* (5.3.12).

The changes for the pseudocode `CODE Exp.1` are given below:

```
CODE Exp.3
Input N, u_0(x), u_L(t), u_R(t), ε, Δt, Tmax
%collocation points, initial data, and c̃_k
Set starting time:  time=0
While time ⩽ Tmax do
  Boundary conditions:  u(1)=u_L(time),  u(N+1)=u_R(time)
  Call the RHS function of the ODE system:  rhs=F(u,N,c̃,ε)
%solve the ODE system, say by using the Euler method
  Set new time level:  time=time+Δt
endwhile
Output u(1),u(2),···,u(N-1)
```

To handle the right-hand side function, the changes for the pseudocode `CODE Exp.2` are given below:

```
CODE Exp.4
function r=F(u,N,c̃,ε)
%calculate coefficients a_k(t)
%calculate coefficients a_k^(i)(t), i=1, 2
%calculate the RHS function of the ODE system
  for j=0 to N do
    r(j)=ε*∑_{k=0}^{N} a(2,k)cos(πjk/N) - u(j)*∑_{k=0}^{N} a(1,k)cos(πjk/N)
  endfor
```

In the following, we list the numerical results with $T_{\max} = 0.5$ and $\epsilon = 1$:

N	$\|e\|_\infty$(Δt=10^{-3})	$\|e\|_\infty$(Δt=10^{-4})	N	$\|e\|_\infty$(Δt=10^{-3})	$\|e\|_\infty$(Δt=10^{-4})
3	3.821e-05	7.698e-05	7	4.689e-05	4.692e-06
4	9.671e-05	5.583e-05	8	4.391e-05	4.393e-06
5	3.644e-05	5.177e-06	9	4.560e-05	4.562e-06
6	4.541e-05	4.376e-06	10	4.716e-05	4.718e-06

It is seen that for $N > 6$ the error is almost unchanged. This again suggests that the error is dominated by that of the time discretization. To fix this problem, we will apply the Runge-Kutta type method discussed in Section 1.6. For the nonlinear Example 5.3.2, applying the collocation spectral methods yields a system of ODEs:

$$\frac{dU}{dt} = \epsilon AU - \text{diag}(U_1, U_2, \cdots, U_{N-1})BU + b(t),$$

where $(B)_{ij} = (D^1)_{ij}, 1 \leqslant i, j \leqslant N-1$, the vector b is associated with the boundary conditions:

$$(b)_j = \epsilon\big[(D^2)_{j,0}U_0 + (D^2)_{j,N}U_N\big] - U_j\big[(D^1)_{j,0}U_0 + (D^1)_{j,N}U_N\big].$$

Note that $U_0 = u_R(t)$ and $U_N = u_L(t)$ are given functions.

In the following, we first give the subroutine for computing the vector b and then give the code which uses the RK4 algorithm (1.6.8) to solve the nonlinear Burgers' equation.

```
CODE RK.2
function b=func_b(N,UL,UR,e_N,D1,D2,U)
for j=1 to N-1 do
  b(j)=ϵ*(D2(j,0)*UR+D2(j,N)*UL)-U(j)*(D1(j,0)*UR+D1(j,N)*UL)
endfor
```

```
CODE RK.3
Input N, u_0(x),u_L(t),u_R(t),ϵ,Δt,Tmax
%Form the matrices A, B and vector b
call CODE DM.3 in Sect 2.1 to get D1(i,j), 0⩽i,j⩽N
D2=D1*D1;
for i=1 to N-1 do
  for j=1 to N-1 do
    A(i,j)=D2(i,j); B(i,j)=D1(i,j)
  endfor
endfor
Set starting time:  time=0
```

```
Set the initial data:  U=u_0(x)
While time ⩽ Tmax do
   %Using RK4 (1.6.8)
   U0=U; C=diag(U(1),U(2),⋯,U(N-1))
   UL=u_L(time); UR=u_R(time);
   b=func_b(N,UL,UR,ϵ,D1,D2,U)
   K1=ϵ*A*U-C*B*U+b
   U=U0+0.5*Δt*K1; C=diag(U(1),U(2),⋯,U(N-1))
   UL=u_L(time+0.5*Δt); UR=u_R(time+0.5*Δt);
   b=func_b(N,UL,UR,ϵ,D1,D2,U)
   K2=ϵ*A*U-C*B*U+b
   U=U0+0.5*Δt*K2; C=diag(U(1),U(2),⋯,U(N-1))
   b=func_b(N,UL,UR,ϵ,D1,D2,U)
   K3=ϵ*A*U-C*B*U+b
   U=U0+Δt*K3; C=diag(U(1),U(2),⋯,U(N-1))
   UL=u_L(time+Δt); UR=u_R(time+Δt);
   b=func_b(N,UL,UR,ϵ,D1,D2,U)
   K4=ϵ*A*U-C*B*U+b
   U=U0+Δt*(K1+2*K2+2*K3+K4)/6
   Set new time level:  time=time+Δt
endwhile
Output U0(1),U(2), ⋯, U(N-1)
```

The maximum errors below are obtained for $\mathrm{T}_{\max} = 0.5$ and $e_N = 1$. The spectral convergence rate is observed for $N = \mathcal{O}(10)$ when RK4 is employed.

N	Max error (Δt=1e-3)	Max error (Δt=5e-4)
3	8.13e-05	8.13e-05
4	5.13e-05	5.13e-05
5	1.82e-06	1.82e-06
6	1.88e-07	1.88e-07
7	7.91e-09	7.91e-09
8	2.19e-09	2.20e-09
9	9.49e-10	8.10e-11

Exercise 5.3

Problem 1 Solve (5.3.1) with the initial condition $u(x,0) = \sin(\pi x)$ by using the Chebyshev spectral methods described in `CODE Exp.1`, except replacing the Euler method by the 2nd-order Runge-Kutta method (1.6.6) with $\alpha = \frac{1}{2}$.

1. Use $N = 6, 8, 9, 10, 11, 12$, and give the maximum errors $|u^N(x,1) - u(x,1)|$.
2. Plot the numerical errors against N using semi-log plot.

Problem 2 Repeat Problem 1, except with the 4th-order Runge-Kutta method (1.6.8).

Problem 3 Solve the problem in Example 5.3.2 by using a pseudo-spectral approach (i.e. using the differential matrix to solve the problem in the physical space). Take $3 \leqslant N \leqslant 20$, and use RK4.

5.4 Fourier spectral methods for the KdV equation

An analytic solution for the KdV equation
Numerical scheme based on a two-step method
Numerical scheme based on the RK4 method
Dual-Petrov Legendre-Galerkin method for the kdv equation

In this section, we describe a method introduced by Fornberg and Whitham [51] to solve the KdV equation

$$u_t + \beta u u_x + \mu u_{xxx} = 0, \quad x \in \mathbb{R}, \tag{5.4.1}$$

where β and μ are given constants. The sign of μ is determined by the direction of the wave and its shape. If $\mu < 0$, by use of the transforms $u \to -u, x \to -x$ and $t \to t$, the KdV equation (5.4.1) becomes

$$u_t + \beta u u_x - \mu u_{xxx} = 0, \quad x \in \mathbb{R}.$$

Therefore, we can always assume $\mu > 0$. Some *linear* properties for solutions of (5.4.1) were observed numerically by Kruskal and Zabusky in 1966[94]. The soliton theory has been motivated by the numerical study of the KdV equation.

In general, the solution of (5.4.1) decays to zero for $|x| \gg 1$. Therefore, numerically we can solve (5.4.1) in a finite domain:

$$u_t + \beta u u_x + \mu u_{xxx} = 0, \qquad x \in (-p,\, p), \tag{5.4.2}$$

with a sufficiently large p.

An analytic solution for the KdV equation

For the computational purpose, it is useful to obtain an exact solution for the nonlinear problem (5.4.1). To this end, we will try to find a traveling wave solution of the form $u(x,t) = V(x - ct)$. Substituting this form into (5.4.1) gives $-cV' + \beta VV' + \mu V''' = 0$, where $V' = V_\zeta(\zeta)$. Integrating once gives

$$-cV + \frac{\beta}{2}V^2 + \mu V'' = \alpha_1,$$

where α_1 is an integration constant. Multiplying the above equation by $2V'$ and integrating again yields

$$-cV^2 + \frac{\beta}{3}V^3 + \mu(V')^2 = 2\alpha_1 V + \alpha_2,$$

where α_2 is a constant. Note that we are looking for the solitary solution: away from the heap of water there is no elevation. This means that $V(x), V'(x), V''(x)$ tend to zero as $|x| \to \infty$, which implies that $\alpha_1 = 0$ and $\alpha_2 = 0$. Therefore, we obtain the ODE

$$-cV^2 + \frac{\beta}{3}V^3 + \mu(V')^2 = 0.$$

One of the solutions for the above nonlinear ODE is

$$V(\zeta) = \frac{3c}{\beta}\text{sech}^2\left(\frac{1}{2}\sqrt{c/\mu}(\zeta - x_0)\right).$$

This can be verified by direct computation. To summarize: One of the exact solutions for the equation (5.4.1) is

$$u(x,t) = \frac{3c}{\beta}\text{sech}^2\left(\frac{1}{2}\sqrt{c/\mu}(x - ct - x_0)\right), \tag{5.4.3}$$

where c and x_0 are some constants.

Numerical scheme based on a two-step method

A simple change of variable ($x \to \pi x/p + \pi$) changes the solution interval $[-p,\ p]$ to $[0, 2\pi]$. The equation (5.4.2) becomes

$$u_t + \frac{\beta\pi}{p}uu_x + \frac{\mu\pi^3}{p^3}u_{xxx} = 0, \qquad x \in [0, 2\pi]. \tag{5.4.4}$$

It follows from (1.5.14) that

$$\frac{\partial^n u}{\partial x^n} = \mathcal{F}^{-1}\{(ik)^n \mathcal{F}\{u\}\}, \qquad n = 1, 2, \cdots.$$

An application of the above results (with $n = 1$ and 3) to (5.4.4) gives

$$\frac{du(x_j,t)}{dt} = -\frac{i\beta\pi}{p}u(x_j,t)F^{-1}(kF(u)) + \frac{i\mu\pi^3}{p^3}F^{-1}(k^3F(u)), \qquad 1 \leqslant j \leqslant N-1, \tag{5.4.5}$$

where we have replaced the continuous Fourier transforms by the discrete transforms. Let $\mathbf{U} = [u(x_1,t), \cdots, u(x_{N-1},t)]^{\mathrm{T}}$. Then (5.4.5) can be written in the vector form

$$\mathbf{U}_t = \mathbf{F}(\mathbf{U}),$$

where $\mathbf{F}$ is defined by (5.4.5). The discretization scheme for time used in [51] is the following two-step method:

$$\mathbf{U}(t+\Delta t) = \mathbf{U}(t-\Delta t) + 2\Delta t\mathbf{F}(\mathbf{U}(t)).$$

For this approach, two levels of initial data are required. The first level is given by the initial function, and the second level can be obtained by using the fourth-order Runge-Kutta method RK4. This way of time discretization for (5.4.5) gives

$$u(x,t+\Delta t) = u(x,t-\Delta t) - 2i\frac{\beta\pi}{p}\Delta t u(x,t)F^{-1}(kF(u)) + 2i\Delta t\frac{\mu\pi^3}{p^3}F^{-1}(k^3F(u)). \tag{5.4.6}$$

Stability analysis for the above scheme gives the stability condition

$$\frac{\Delta t}{\Delta x^3} < \frac{1}{\pi^3} \approx 0.0323. \tag{5.4.7}$$

In order that the FFT algorithms can be applied directly, we use the transform $k' = k + N/2$ in (1.5.15) and (1.5.16) to obtain

$$\hat{u}(k',t) = F(u) = \frac{1}{N}\sum_{j=0}^{N-1}(-1)^j u(x_j,t)e^{-ik'x_j}, \qquad 0 \leqslant k' \leqslant N-1,$$

$$F^{-1}(kF(u)) = (-1)^j \sum_{k'=0}^{N-1}(k'-N/2)\hat{u}(k',t)e^{ik'x_j}, \qquad 0 \leqslant j \leqslant N-1.$$

$$F^{-1}(k^3F(u)) = (-1)^j \sum_{k'=0}^{N-1}(k'-N/2)^3\hat{u}(k',t)e^{ik'x_j}, \qquad 0 \leqslant j \leqslant N-1.$$

The FFT algorithms introduced in Section 1.5 can then be used. A pseudo-code is given below:

```
CODE KdV.1
```

```
Input β, p, μ, N, u0(x), λ (Δt=λΔx^3)
Δx=2*p/N;  Δt=λΔx^3
Grid points x(j) and initial data:  x(j)=2πj/N; u0(j)=u0(j)
% use an one-step method to compute u1(j):=u(xj,Δt)
time=Δt
while time < T do
    for j=0 to N-1 do
       %Need to call function to calculate F(u1)
       RHS=F(u1,N,β, μ, p);  u(j)=u0(j)+2Δt*RHS(j)
    endfor
%update vectors u0 and u1
    for j=0 to N-1 do
       u0(j)=u1(j);  u1(j)=u(j)
    endfor
 Update time:  time=time+Δt
 endwhile
 Output the solution u(j) which is an approximation to u(xj,T).
```

In forming the function $\mathbf{F}$(u1,N,β, μ, p), CODE FFT.1 and CODE FFT.2 introduced in Section 1.5 can be used. If the programs are written in MATLAB, the MATLAB functions fft and ifft can be used directly. In MATLAB, fft(x) is the discrete Fourier transform of vector x, computed with a fast Fourier algorithm. X=fft(x) and x= ifft(X) implement the transform and the inverse transform pair given for vectors of length N by

$$X(k) = \sum_{j=1}^{N} x(j) e^{-2\pi i (j-1)(k-1)/N}, \quad x(j) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{2\pi i (j-1)(k-1)/N}. \tag{5.4.8}$$

Note that $X(k)$ has an N^{-1} factor difference with our definition in Section 1.5. With the above definitions, a pseudo-code for the function $\mathbf{F}(u, N, \beta, \mu, p)$ used above is given below:

```
CODE KdV.2
function r=F(u,N,β, μ, p)
for j=0 to N-1 do
   y(j)=(-1)^j*u(j)
endfor
Compute F(u):  Fu=fft(y)/N
%compute kF(u) and k^3F(u)
for k=0 to N-1 do
   y(k)=(k-N/2)*Fu(k);  Fu(k)=(k-N/2)^3Fu(k)
endfor
```

```
Compute the two inverses in the formula:  y=ifft(y)*N;
Fu=ifft(Fu)*N
%compute F(u,N,β, μ, p)
for j=0 to N-1 do
   r(j)=-i*β*π/p*u(j)*(-1)^j*u(j)+i*μ*(π/p)^3*(-1)^j*Fu(j)
endfor
```

Since `fft` and `ifft` are linear operators, it can be verified that without the N factors in the steps involving `fft` and `ifft` we will end up with the same solutions. The MATLAB code for the function $\mathbf{F}(u, N, \beta, \mu, p)$, `CODE KdV.3`, can be found in this book's website:

`http://www.math.hkbu.edu.hk/~ttang/PGteaching`

Example 5.4.1 *Consider the KdV equation* (5.4.1) *with* $\beta = 6$, $\mu = 1$, *and initial condition* $u(x, 0) = 2\text{sech}^2(x)$.

By (5.4.3), the exact solution for this problem is $u(x, t) = 2\text{sech}^2(x - 4t)$. The main program written in MATLAB for computing $u(x, 1)$, `CODE KdV.4`, can also be found in this book's website, where for simplicity we assume that $u(x, \Delta t)$ has been obtained exactly.

In Figure 5.1, we plot the exact solution and numerical solutions at $t = 1$ with $N = 64$ and 128. The spectral convergence is observed from the plots.

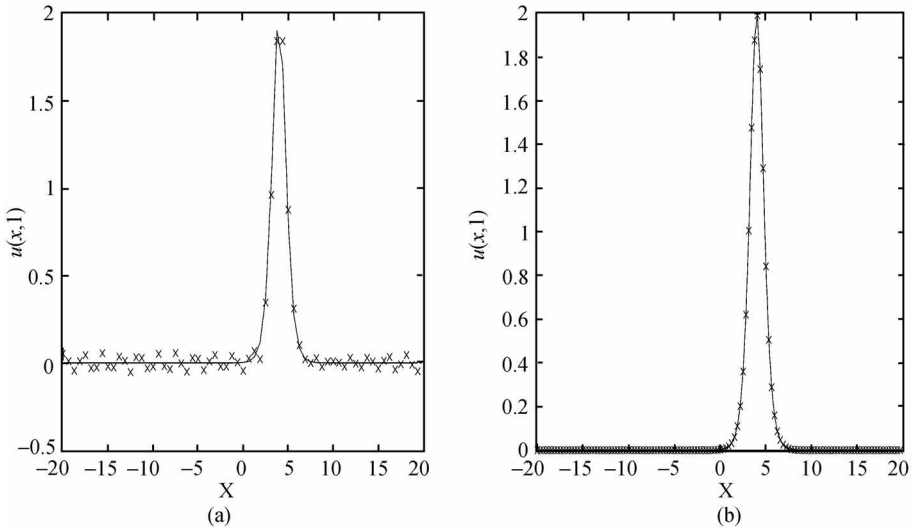

Figure 5.1

Fourier spectral solution of the KdV equation with (a): *N*=64(the maximum error is 1.32e-0.1), and (b): *N*=128(the maximum error is 6.08e-0.3).

Fornberg and Whitham modified the last term of (5.4.6) and obtained the following scheme

$$u(x,t+\Delta t) = u(x,t-\Delta t) - 2i\frac{\beta\pi}{p}\Delta t u(x,t)F^{-1}(kF(u)) + 2iF^{-1}\left(\sin\left(\mu\pi^3 k^3 p^{-3}\Delta t\right)F(u)\right). \tag{5.4.9}$$

Numerical experiments indicate that (5.4.9) requires less computing time than that of (5.4.6). The stability condition for this scheme is

$$\frac{\Delta t}{\Delta x^3} < \frac{3}{2\pi^3} \approx 0.0484, \tag{5.4.10}$$

which is an improvement of (5.4.7).

Numerical scheme based on the RK4 method

Numerical experiments suggest that the stability condition can be improved by using RK4 introduced in Section 1.6. A modified code using the formula (1.6.11), `CODE KdV.5`, can be found in the website of this book. The errors with $N = 64$, 128 and 256 are listed below:

N	Maximum error	time step
64	8.65e-02	1.59e-02
128	1.12e-05	1.98e-03
256	1.86e-12	2.48e-04

It is observed that comparing with the second-order time stepping methods (5.4.6) and (5.4.9) the RK4 method allows larger time steps and leads to more accurate numerical approximations.

Dual-Petrov Legendre-Galerkin method for the KdV equation

Although most of the studies on the KdV equation are concerned with initial value problems or initial value problems with periodic boundary conditions as we addressed in the previous section, it is often useful to consider the KdV equation on a semi-infinite interval or a bounded interval. Here, as an example of application to nonlinear equations, we consider the KdV equation on a finite interval:

$$\begin{aligned} &\alpha v_t + \beta v_x + \gamma v v_x + v_{xxx} = 0, \quad x \in (-1,1),\ t \in (0,T], \\ &v(-1,t) = g(t),\ v(1,t) = v_x(1,t) = 0, \quad t \in [0,T], \\ &v(x,0) = v_0(x), \quad x \in (-1,1). \end{aligned} \tag{5.4.11}$$

The positive constants α, β and γ are introduced to accommodate the scaling of the spatial interval. The existence and uniqueness of the solution for (5.4.11) can be established as in [32],[16]. Besides its own interests, the equation (5.4.11) can also be viewed as a legitimate approximate model for the KdV equation on a quarter-plane before the wave reaches the right boundary.

Let us first reformulate (5.4.11) as an equivalent problem with homogeneous boundary conditions. To this end, let $\hat{v}(x,t) = \frac{1}{4}(1-x)^2 g(t)$ and write $v(x,t) = u(x,t) + \hat{v}(x,t)$. Then, u satisfies the following equation with homogeneous boundary conditions:

$$\begin{aligned} &\alpha u_t + a(x,t)u + b(x,t)u_x + \gamma u u_x + u_{xxx} = f, \quad x \in (-1,1),\ t \in (0,T], \\ &u(\pm 1, t) = u_x(1,t) = 0, \quad t \in [0,T], \\ &u(x,0) = u_0(x) = v_0(x) - \hat{v}(x,0), \quad x \in (-1,1), \end{aligned} \tag{5.4.12}$$

where

$$a(x,t) = \frac{\gamma}{2}(x-1)g(t), \quad b(x,t) = \beta + \gamma \hat{v}(x,t), \quad f(x,t) = -\alpha \hat{v}_t(x,t).$$

We consider the following Crank-Nicolson leap-frog dual-Petrov-Galerkin approximation:

$$\begin{aligned} &\frac{\alpha}{2\Delta t}(u_N^{k+1} - u_N^{k-1}, v_N)_{\omega^{-1,1}} + \frac{1}{2}(\partial_x(u_N^{k+1} + u_N^{k-1}), \partial_x^2(v_N \omega^{-1,1})) \\ &= (I_N f(\cdot, t_k), v_N)_{\omega^{-1,1}} - \gamma (I_N(u_N^k \partial_x u_N^k), v_N)_{\omega^{-1,1}} \\ &\quad - (a\, u_N^k, v_N)_{\omega - 11} + (u_N^k, \partial_x(b\, v_N\, \omega^{-1,1})), \quad \forall v_N \in V_N. \end{aligned} \tag{5.4.13}$$

This is a second-order (in time) two-step method so we need to use another scheme, for example, the semi-implicit first-order scheme, to compute u_N^1. Since the truncation error of a first-order scheme is $\mathcal{O}(\Delta t)^2$, the overall accuracy of the scheme (5.4.13) will still be second-order in time.

It is shown in [145] that this scheme is stable under the very mild condition $\delta N \leqslant C$ (as opposed to $\delta N^3 \leqslant C$ in the previous subsections). Setting $u_N = \frac{1}{2}(u_N^{k+1} - u_N^{k-1})$, we find at each time step we need to solve

$$\frac{\alpha}{2\Delta t}(u_N, v_N)_{\omega^{-1,1}} + (\partial_x u_N, \partial_x^2(v_N \omega^{-1,1})) = (h, v_N)_{\omega^{-1,1}}, \quad \forall v_N \in V_N, \tag{5.4.14}$$

where h includes all the known terms from previous time steps. The system (5.4.14)

is exactly in the form of (3.7.27) for which a very efficient algorithm is presented in Section 3.6. Note that the nonlinear term $I_N(u_N \partial_x u_N)$ can be computed by a procedure described in Section 2.5.

Now, we present some numerical tests for the KdV equation. We first consider the initial value KdV problem

$$u_t + uu_x + u_{xxx} = 0, \quad u(x,0) = u_0(x), \tag{5.4.15}$$

with the exact soliton solution

$$u(x,t) = 12\kappa^2 \text{sech}^2(\kappa(x - 4\kappa^2 t - x_0)). \tag{5.4.16}$$

Since $u(x,t)$ converges to 0 exponentially as $|x| \to \infty$, we can approximate the initial value problem (5.4.15) by an initial boundary value problem for $x \in (-M, M)$ as long as the soliton does not reach the boundaries.

We take $\kappa = 0.3$, $x_0 = -20$, $M = 50$ and $\Delta t = 0.001$ so that for $N \lesssim 160$, the time discretization error is negligible compared with the spatial discretization error. In Figure 5.2, we plot the time evolution of the exact solution, and in Figure 5.3, we plot the maximum errors in the semi-log scale at $t = 1$ and $t = 50$. Note that the straight lines indicate that the errors converge like e^{-cN} which is typical for solutions that are infinitely differentiable but not analytic. The excellent accuracy for this known exact solution indicates that the KdV equation on a finite interval can be used to effectively simulate the KdV equation on a semi-infinite interval before the wave reaches the boundary.

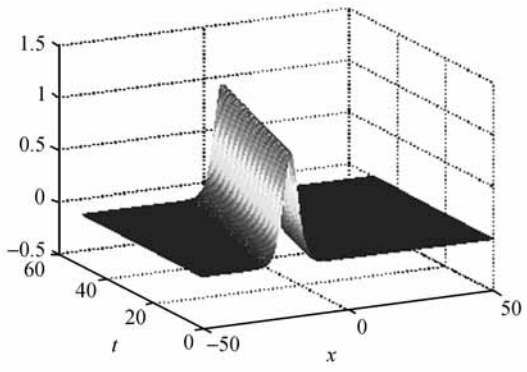

Figure 5.2 Time evolution for exact KdV solution (5.4.16)

In the following tests, we fix $M = 150$, $\Delta t = 0.02$ and $N = 256$. We start with the following initial condition

$$u_0(x) = \sum_{i=1}^{5} 12\kappa_i^2 \operatorname{sech}^2(\kappa_i(x - x_i)) \tag{5.4.17}$$

with

$$\begin{aligned} &\kappa_1 = 0.3,\ \kappa_2 = 0.25,\ \kappa_3 = 0.2,\ \kappa_4 = 0.15,\ \kappa_5 = 0.1, \\ &x_1 = -120,\ \ x_2 = -90,\ \ x_3 = -60,\ x_4 = -30,\ \ x_5 = 0. \end{aligned} \tag{5.4.18}$$

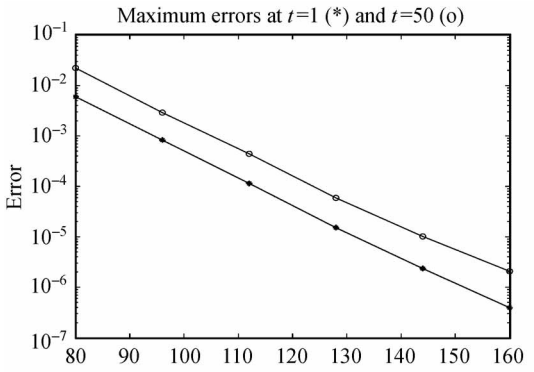

Figure 5.3 The KdV problem (5.4.15) and (5.4.16): maximum error vs. N.

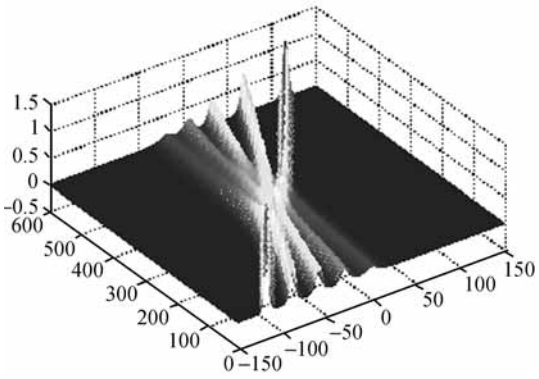

Figure 5.4 Time evolution for the numerical solution to (5.4.15) and (5.4.17).

In Figure 5.4, we plot the time evolution of the solution in the (x, t) plane. We also plot the initial profile and the profile at the final step ($t = 600$) in Figure 5.5. We observe that the soliton with higher amplitude travels with faster speed, and the amplitudes of the five solitary waves are well preserved at the final time. This indicates that our scheme has an excellent conservation property.

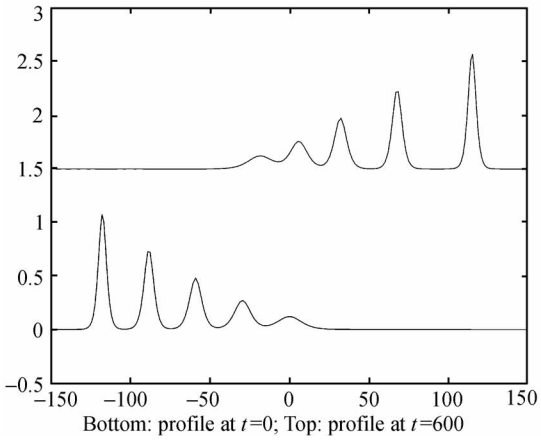

Figure 5.5 Top curve is the initial profile (5.4.17) and the bottom is the profile at $t = 600$.

Exercise 5.4

Problem 1 Consider the Sine-Gordon equation,

$$u_{tt} = u_{xx} - \sin(u), \qquad x \in (-\infty, \infty), \tag{5.4.19}$$

with initial conditions

$$u(x,0) = 0, \qquad u_t(x,0) = 2\sqrt{2}\text{sech}(x/\sqrt{2}). \tag{5.4.20}$$

The exact solution for this problem is

$$u(x,t) = 4\tan^{-1}\left(\frac{\sin(t/\sqrt{2})}{\cosh(x/\sqrt{2})}\right).$$

The Sine-Gordon equation (5.4.19) is related to the KdV and cubic Schrödinger equations in the sense that all these equations admit soliton solutions.

1. Reduce the second-order equation (5.4.19) by introducing the auxiliary variable u_t.

2. Applying the method used in this section to solve the above problem in a truncated domain $(-12.4, 12.4)$, with $N = 32$ and 64. Show the maximum absolute errors at $t = 2\pi, 4\pi$ and 6π.

3. The numerical solution in the (x,t) plane is plotted in Figure 5.6. Verify it with your numerical solution.

Problem 2 Use the Fourier spectral methods to solve the Burgers' equation (5.3.11) with $\epsilon = 0.15$ and with periodic boundary condition in $[-\pi, \pi]$ (p. 113, [165]). The initial data is

$$u(x,0) = \begin{cases} \sin^2 x, & \text{for } x \in [-\pi, 0], \\ 0, & \text{for } x \in (0, \pi]. \end{cases}$$

Produce solution plots at time $0, 0.5, 1, \cdots, 3$, with a sufficiently small time step, for $N = 64, 128$ and 256. For $N = 256$, how small a value of ϵ can you take without obtaining unphysical oscillations?

Problem 3 Write a spectral code to solve the following nonlinear Schrödinger's equation for super-fluids:

$$\begin{aligned} &i\epsilon u_t + \epsilon^2 u_{xx} + (|u|^2 - 1)u = 0, \quad x \in (-\pi, \pi), \\ &u(x,0) = x^2 e^{-2x^2} e^{i}, \end{aligned}$$

where $\epsilon = 0.3$. The problem has a periodic boundary condition in $[-\pi, \pi]$. Choose a proper time stepping method to solve the problem for $0 < t \leqslant 8$.

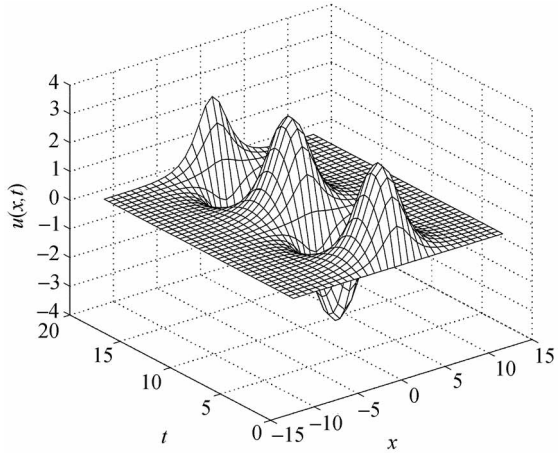

Figure 5.6 Breather solution of the sine-Gordon equation.

5.5 Fourier method and filters

Fourier approximation to discontinuous function
Spectral-filter
Fully discretized spectral-Fourier method

We consider what is accepted by now as the universal model problem for scalar conservation laws, namely, the inviscid Burgers' equation

$$u_t + \left(u^2/2\right)_x = 0, \tag{5.5.1}$$

subject to given initial data. We want to solve the 2π-periodic problem (5.5.1) by the spectral-Fourier method. To this end, we approximate the spectral-Fourier projection of $u(x,t)$,

$$P_N u = \sum_{k=-N}^{N} \hat{u}_k e^{ikx}, \qquad \hat{u}_k = \frac{1}{2\pi}\int_0^{2\pi} u(x)e^{-ikx}\mathrm{d}x, \tag{5.5.2}$$

by an N-trigonometric polynomial, $u^N(x,t)$,

$$u^N(x,t) = \sum_{k=-N}^{N} \hat{u}_k(t)e^{ikx}. \tag{5.5.3}$$

In this method the fundamental unknowns are the coefficients $\hat{u}_k(t)$, $|k| \leqslant N$. A set of ODEs for the $\hat{u}_k$ are obtained by requiring that the residual of (5.5.1) be orthogonal to all the test functions e^{-ikx}, $|k| \leqslant N$:

$$\int_0^{2\pi} (u_t^N + u^N\, u_x^N)e^{-ikx}\,\mathrm{d}x = 0.$$

Due to the orthogonality property of the test and trial functions,

$$\hat{u}_k'(t) + \widehat{(u^N u_x^N)}_k = 0, \quad |k| \leqslant N, \tag{5.5.4}$$

where

$$\widehat{(u^N\, u_x^N)}_k = \frac{1}{2\pi}\int_0^{2\pi} u^N\, u_x^N e^{-ikx}\,\mathrm{d}x. \tag{5.5.5}$$

The initial condition are clearly

$$\hat{u}_k(0) = \frac{1}{2\pi}\int_0^{2\pi} u(x,0)e^{-ikx}\,\mathrm{d}x. \tag{5.5.6}$$

Equation (5.5.5) is a particular case of the general quadratic nonlinear term

$$\widehat{(uv)}_k = \frac{1}{2\pi}\int_0^{2\pi} uve^{-ikx}\mathrm{d}x, \tag{5.5.7}$$

where u and v denote generic trigonometric polynomials of degree N, which have expansions similar to (5.5.2). When these are inserted into (5.5.7) and the orthogo-

nality property is invoked, the expression

$$\widehat{(uv)}_k = \sum_{p+q=k} \hat{u}_p \hat{v}_q \tag{5.5.8}$$

results. The ODE system (5.5.4) is discretized in time by an ODE solver such as Runge-Kutta methods described in Section 1.6.

Fourier approximation to discontinuous function

The Fourier approximation (5.5.3) is a very good way of reconstructing the point values of $u(x,t)$ provided that u is smooth and periodic. However, if a *discontinuous* function $u(x,t)$ is approximated by its finite Fourier series $P_N u$, then the order of convergence of $P_N u$ to u is only $\mathcal{O}(N^{-1})$ for each fixed point[62, 63]. Moreover, $P_N u$ has *oscillations* of order 1 in a neighborhood of $\mathcal{O}(N^{-1})$ of the discontinuity. To see this, we consider a simple test problem.

Example 5.5.1 *Consider the Burgers' equation* (5.5.1) *with the initial data*

$$u(x) = \begin{cases} \sin(x/2), & 0 \leqslant x \leqslant 1.9, \\ -\sin(x/2), & 1.9 < x < 2\pi. \end{cases} \tag{5.5.9}$$

Figure 5.7 shows the behavior of the spectral-Fourier solution for the Burgers' equation, which is subject to the discontinuous initial condition (5.5.9). The resulting ODE system for the Fourier coefficients was integrated up to $t = 0.1$ using the third-order Runge-Kutta method. The oscillatory behavior of the numerical solution is clearly observed from this figure.

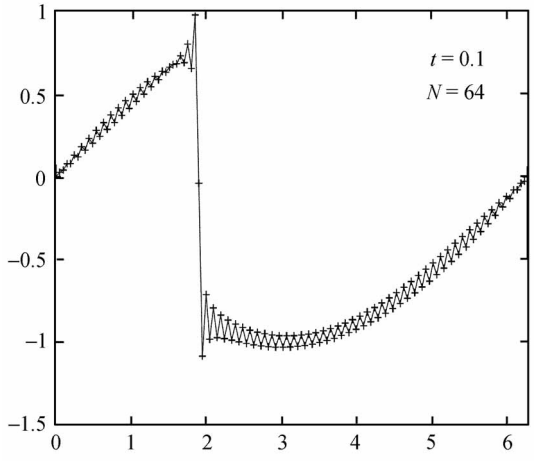

Figure 5.7 Spectral-Fourier solution with $N = 64$.

Spectral-filter

There have been many attempts to smooth the oscillatory solution. It was observed that the oscillations may be suppressed, or smoothed, by a gradual tapering of the Fourier coefficients. Among them is a *spectral-filter* approach in which the first step is to solve for the coefficients of the spectral expansion, $\hat{u}_k$, and then to multiply the resulting coefficient by a factor $\sigma_k = \sigma(k/N)$. Here σ is called a filter. We will follow the presentation by Vandeven [166] and Gottlieb and Shu[63] to introduce the Fourier space filter of order p. A *real and even* function $\sigma(\eta)$ is called a filter of order p if

- (i) $\sigma(0) = 1,\ \sigma^{(l)}(0) = 0, 1 \leqslant l \leqslant p-1.$
- (ii) $\sigma(\eta) = 0, \quad |\eta| \geqslant 1.$
- (iii) $\sigma(\eta) \in C^{(p-1)}, \quad |\eta| < \infty\,.$

There are many examples of filters that have been used during the years. We would like to mention some of them:

• In 1900, Fejér suggested using averaged partial sums instead of the original sums. This is equivalent to the first order filter

$$\sigma_1(\eta) = 1 - \eta^2.$$

• The Lanczos filter is formally a first-order one,

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

However, note that at $\eta = 0$, it satisfies the condition for a second-order filter.

• A second-order filter is the raised cosine filter

$$\sigma_3(\eta) = 0.5(1 + \cos(\pi\eta)).$$

• The sharpened raised cosine filter is given by

$$\sigma_4(\eta) = \sigma_3^4(\eta)(35 - 84\sigma_3(\eta) + 70\sigma_3^2(\eta) - 20\sigma_3^3(\eta)).$$

This is an eighth-order filter.

• The exponential filter of order p (for even p) is given by

$$\sigma_5(\eta) = e^{-\alpha\eta^p}\,.$$

Note that formally the exponential filter does not conform to the definition of the filter as $\sigma_5(1) = e^{-\alpha}$. However, in practice we choose α such that $e^{-\alpha}$ is within the round-off error of the specific computer.

The filtering procedures may be classified as follows:

• **Pre-processing** The initial condition is filtered in terms of its continuous Fourier coefficients

$$u_0^{new}(x) = \sum_{k=-N}^{N} \sigma(2\pi k/N)\hat{u}_k(0)e^{ikx}.$$

• **Derivative filtering** In the computation of spatial derivatives the term ik is replaced by $ik\sigma(2\pi k/N)$, i.e.,

$$\frac{du}{dx} = \sum_{k=-N}^{N} ik\sigma(2\pi k/N)\hat{u}_k e^{ikx}.$$

• **Solution smoothing** At regular intervals in the course of advancing the solution in time, the current solution values are smoothed in Fourier space, i.e.,

$$u(x,t) = \sum_{k=-N}^{N} \sigma(2\pi k/N)\hat{u}_k(t)e^{ikx}.$$

Fully discretized spectral-Fourier method

In order to illustrate the use of the spectral-filter approach, we will discuss a more general fully discretized method which uses the standard spectral-Fourier method in space and Runge-Kutta methods in time. This fully discretized method will also be employed in the next section. Consider the conservation equation

$$\begin{aligned} &u_t + f(u)_x = 0, \quad 0 \leqslant x < 2\pi, \quad t > 0, \\ &u(x,0) = u_0(x), \quad 0 \leqslant x < 2\pi. \end{aligned} \tag{5.5.10}$$

If the *cell average* of u is defined by

$$\bar{u}(x,t) = \frac{1}{\Delta x}\int_{x-\Delta/2}^{x+\Delta x/2} u(\zeta,t)\mathrm{d}\zeta\,, \tag{5.5.11}$$

then (5.5.10) can be approximated by

$$\begin{aligned} &\frac{\partial}{\partial t}\bar{u}(x,t) + \frac{1}{\Delta x}\Big[f\left(u(x+\Delta x/2,t)\right) - f\left(u(x-\Delta x/2,t)\right)\Big] = 0, \\ &\bar{u}(x,0) = \bar{u}^0(x). \end{aligned} \tag{5.5.12}$$

Hence a semi-discrete conservative scheme

$$\frac{d}{dt}\bar{u}_j = L(\bar{u})_j := -\frac{1}{\Delta x}\left(\hat{f}_{j+1/2} - \hat{f}_{j-1/2}\right) \tag{5.5.13}$$

will be of high order if the numerical flux $\hat{f}_{j+1/2}$ approximates $f\left(u(x+\Delta x/2,t)\right)$ to high order. Notice that (5.5.13) is a scheme for the cell averages $\bar{u}_j$. However, in evaluating $\hat{f}_{j+1/2}$, which should approximate $f(u(x_j+\Delta x/2,t))$, we also need accurate point values $u_{j+1/2}$. For finite difference schemes the reconstruction from cell averages to point values is a major issue and causes difficulties. For spectral methods, this is very simple because $\bar{u}$ is just the convolution of u with the characteristic function of $(x_{j-1/2}, x_{j+1/2})$. To be specific, if

$$u(x) = \sum_{k=-N}^{N} a_k e^{ikx} \tag{5.5.14}$$

(we have suppressed the time variable t), then

$$\bar{u}(x) = \sum_{k=-N}^{N} \bar{a}_k e^{ikx} \tag{5.5.15}$$

with

$$\bar{a}_k = \tau_k a_k\,, \qquad \tau_k = \frac{\sin(k\Delta x/2)}{k\Delta x/2} \quad \text{for } 0 < |k| \leqslant N,\ \tau_0 = 1. \tag{5.5.16}$$

We now state our scheme as (5.5.13) with

$$\hat{f}_{j+1/2} = f(u(x_{j+1/2},t)), \tag{5.5.17}$$

where u is defined by (5.5.14). We obtain the Fourier coefficients $\bar{a}_k$ of $\bar{u}$ from $\{\bar{u}_j\}$ by collocation, and obtain a_k of u needed in (5.5.14) by (5.5.16). To discretize (5.5.13) in time, we use the high-order TVD Runge-Kutta methods proposed in [148]:

$$\begin{aligned} \bar{u}^{(j)} &= \sum_{k=0}^{j-1} \left(\alpha_{jk} \bar{u}^{(k)} + \beta_{jk} \Delta t L(\bar{u}^{(k)}) \right), \quad j = 1, \cdots, r, \\ \bar{u}^{(0)} &= \bar{u}^n, \quad \bar{u}^{n+1} = \bar{u}^{(r)}. \end{aligned} \tag{5.5.18}$$

In our computation, we will use a third-order Runge-Kutta scheme, i.e. $r = 3$, with $\alpha_{10} = \beta_{10} = 1, \alpha_{20} = \frac{3}{4}, \beta_{20} = 0, \alpha_{21} = \beta_{21} = \frac{1}{4}, \alpha_{30} = \frac{1}{3}, \beta_{30} = \alpha_{31} = \beta_{31} = 0, \alpha_{32} = \beta_{32} = \frac{2}{3}$. A small Δt will be used so that the temporal error can be neglected. A suggested algorithm can be

• (1) Starting with $\{\bar{u}_j\}$, compute its collocation Fourier coefficients $\{\bar{a}_k\}$ and Fourier coefficients $\{a_k\}$ of u by (5.5.16).

• (2) Compute $u(x_{j+1/2})$ by

$$u(x) = \sum_{k=-N}^{N} \sigma(2\pi k/N) a_k e^{ikx}. \tag{5.5.19}$$

In the above, a *solution smoothing* with a filter function is used.

• (3) Use $\hat{f}_{j+1/2} = f(u(x_{j+1/2}, t))$ in (5.5.13), and use the third-order Runge-Kutta method (5.5.18).

• (4) After the last time step, use a stronger filter (i.e. lower order filter) in (5.5.19) to modify the numerical solution $u(x, T)$.

A pseudocode outlining the above procedure is provided below:

```
CODE Shock.1
Input N, α_jk and β_jk, 1⩽j⩽3, 0⩽k⩽2
Input Δt, u_0(x), T
      Δx=2π/(2N+1), x_j=jΔx  0⩽j⩽2N
Compute ū(j,0)=ū_0(x_j), |j|⩽2N, using (5.5.11)
time=0
While time < T
 For r=0 to 2 do
  %Compute Fourier coefficients of ū and u using collocation
  method
  for |k|⩽ N do
       ā_k= (∑_{j=0}^{2N} ū(j,r) e^{-ikx_j})/(2N+1)
       τ_k=sin(kΔx/2)/(kΔx/2); a_k=ā_k/τ_k
    endfor
  %Compute u(x_{j+1/2}) using a weak filter
    for j=0 to 2N do
```

```
        x_{j+1/2}=x_j+0.5*Δx; u(x_{j+1/2})=Σ_{k=-N}^{N} σ(2πk/N) a_k e^{ikx_{j+1/2}}
    endfor
   %Runge-Kutta method
    for j=0 to 2N do
       RHS(j,r)=-(f(u(x_{j+1/2}))-f(u(x_{j-1/2})))/Δx
       if r=0 then
          ū(j,1)=ū(j,0)+Δt RHS(j,0)
          elseif r=1 then
          ū(j,2)=(3/4)ū(j,0)+(1/4)ū(j,1)+(Δt/4)RHS(j,1)
          elseif r=2 then
          ū(j,3)=(1/3)ū(j,0)+(2/3)ū(j,2)+(2/3)ΔtRHS(j,2)
       endif
    endfor
 endFor
Update the initial value:  ū(j,0)=ū(j,3), 0⩽j⩽2N
time=time+Δt
endWhile
 %Final solution smoothing using a stronger filter σ̃
   for |k|⩽ N do
       ā_k=(Σ_{j=0}^{2N} ū_j e^{-ikx_j})/(2N+1); a_k=ā_k/τ_k
   endfor
   for j=0 to 2N do
      u(x_j)=Σ_{k=-N}^{N} σ̃(2πk/N) a_k e^{ikx_{j+1/2}}
   endfor
```

We reconsider Example 5.5.1 by using `CODE Shock.1`. The weak filter σ used above is

$$\sigma(\eta) = e^{-15\ln 10\eta^{20}} \tag{5.5.20}$$

and the strong filter $\tilde{\sigma}$ used above is

$$\sigma(\eta) = e^{-15\ln 10\eta^{4}}. \tag{5.5.21}$$

The time step used is $\Delta t = 0.01$ and $T = 0.5$. The filtered spectral-Fourier solution with $N = 64$ is displayed in Figure 5.8, which is an improvement to the standard spectral-Fourier solution.

Exercise 5.5

Problem 1 Solve the periodic Burgers' equation

$$\begin{aligned} &u_t + (u^2/2)_x = 0, \quad x \in [0, 2\pi],\ t > 0 \\ &u(x,0) = \sin(x), \quad u(0,t) = u(2\pi,t), \end{aligned} \tag{5.5.22}$$

using (a) the spectral-Fourier method; and (b) the filtered spectral-Fourier method.

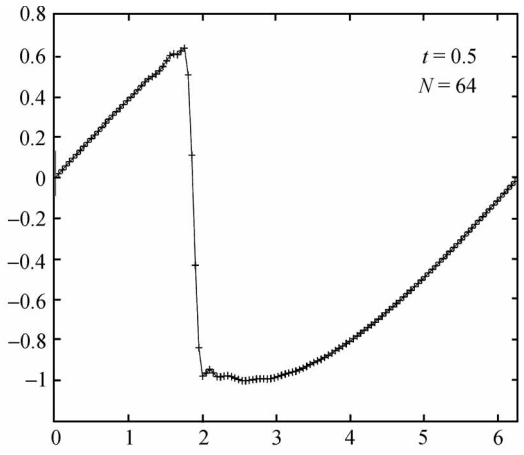

Figure 5.8 Filtered spectral-Fourier solution with $N = 64$.

5.6 Essentially non-oscillatory spectral schemes

Spectral accuracy from the Fourier coefficients
Essentially non-oscillatory reconstruction

Naive implementations of the spectral method on hyperbolic problems with discontinuous solutions will generally produce oscillatory numerical results. The oscillations arising directly from the discontinuity have a Gibbs-like, high-frequency character. These oscillations are not in themselves insurmountable, for according to a result of Lax[96], they should contain sufficient information to permit the reconstruction of the correct physical solution from the visually disturbing numerical one.

We consider the one-dimensional scalar conservations law

$$u_t + f(u)_x = 0, \tag{5.6.1}$$

with prescribed initial conditions, $u(x, 0) = u_0(x)$. It is well known that solutions of (5.6.1) may develop spontaneous jump discontinuities (shock waves) and hence the class of weak solutions must be admitted. Moreover, since there are many possible weak solutions, the equation (5.6.1) is augmented with an entropy condition which requires

$$U(u)_t + F(u)_x \leqslant 0. \tag{5.6.2}$$

Here, $U(u)$ and $F(u)$ are any entropy function and the corresponding entropy-flux pair associated with (5.6.1), so that a strict inequality in (5.6.2) reflects the existence of physical relevant shock waves in the entropy solution of (5.6.1) and (5.6.2). Further theoretical support for the use of spectral methods on non-smooth problems was furnished by many authors, see e.g. [59], [157], [115], [73]. Lax and Wendroff[97] proved that if the sequence $u^N (N \geqslant 0)$ of the solutions produced by a Fourier or Chebyshev collocation method for the equation (5.6.1) is bounded and converges almost everywhere, as $N \to \infty$, then the limit is a weak solution of (5.6.1). This means that it satisfies the equation

$$\iint (u\phi_t + f(u)\,\phi_x)\,\mathrm{d}x\,\mathrm{d}t = \int u(x,0)\,\phi(x,0)\,\mathrm{d}x$$

for all smooth functions ϕ which vanish for large t and on the boundary of the domain. The limit solution thus satisfies the jump condition, $s = [f(u)]/[u]$. Hence, any shocks that are present are propagated with the correct speed.

It was observed by many authors that using a filter approach is equivalent to adding an artificial viscosity for finite difference methods. When applied too often, a strong filter will unacceptably *smear* out a shock. On the other hand, frequent applications of a weak filter may not be enough even to stabilize the calculation. In this section, we follow Cai, Gottlieb and Shu[28] to describe a Fourier spectral method for shock wave calculations.

Spectral accuracy from the Fourier coefficients

For simplicity, assume that $u(x)$, $0 \leqslant x \leqslant 2\pi$, is a periodic piecewise smooth function with *only one point of discontinuity* at $x = \alpha$, and denote by $[u]$ the value of the jump of $u(x)$ at α, namely $[u] = (u(\alpha^+) - u(\alpha^-))/2\pi$. We assume that the first $2N+1$ Fourier coefficients $\hat{u}_k$ of $u(x)$ are known and given by (5.5.2). The objective is to construct an *essentially non-oscillatory* spectrally accurate approximation to $u(x)$ from the Fourier coefficients $\hat{u}'_k s$. We start by noting that the Fourier coefficients $\hat{u}'_k s$ contain information about the shock position α and the magnitude $[u]$ of the shock:

Lemma 5.6.1 *Let u be a periodic piecewise smooth function with one point of discontinuity α. Then for $|k| \geqslant 1$ and for any $n > 0$,*

$$\hat{u}_k = e^{-ik\alpha} \sum_{j=0}^{n-1} \frac{[u^{(j)}]}{(ik)^{j+1}} + \frac{1}{2\pi} \int_0^{2\pi} \frac{[u^{(n)}]}{(ik)^n} e^{-ikx} \mathrm{d}x\,. \tag{5.6.3}$$

Proof It follows from

$$\hat{u}_l = \frac{1}{2\pi}\int_0^{2\pi} u(x)e^{-ilx}\,\mathrm{d}x = \frac{1}{2\pi}\int_0^{x_s} u(x)e^{-ilx}\,\mathrm{d}x + \frac{1}{2\pi}\int_{x_s}^{2\pi} u(x)e^{-ilx}\,\mathrm{d}x,$$

and integration by parts that

$$\hat{u}_l = e^{-ilx_s}\frac{u(x_s^+) - u(x_s^-)}{2\pi il} + \frac{1}{2\pi}\int_0^{2\pi}\frac{u'(x)e^{-ilx}}{il}\,\mathrm{d}x;$$

the rest is obtained by induction. This completes the proof. □

As an example, consider the sawtooth function $F(x,\alpha,A)$ defined by

$$F(x,\alpha,A) = \begin{cases} -Ax, & x \leqslant \alpha\,, \\ A(2\pi - x), & x > \alpha\,. \end{cases} \tag{5.6.4}$$

Note that the jump of the function, $[F]$, is A and all the derivatives are continuous: $[F^{(j)}] = 0$ for all $j \geqslant 1$. That means the expansion (5.6.3) can be terminated after the first term, yielding the following results for $\hat{f}_k$, the Fourier coefficients of $F(x,\alpha,A)$:

$$\hat{f}_0(\alpha,A) = A(\pi - \alpha), \quad \hat{f}_k(\alpha,A) = \frac{Ae^{-ik\alpha}}{ik}\,, \quad |k| \geqslant 1\,. \tag{5.6.5}$$

This example suggests that we can rewrite (5.6.3) as

$$\hat{u}_k = \hat{f}_k(\alpha,[u]) + e^{-ik\alpha}\sum_{j=1}^{n-1}\frac{[u^{(j)}]}{(ik)^{j+1}} + \frac{1}{2\pi}\int_0^{2\pi}\frac{[u^{(n)}]}{(ik)^n}e^{-ikx}\mathrm{d}x\,, \quad |k| \geqslant 1\,. \tag{5.6.6}$$

The order one oscillations in approximating $u(x)$ by its finite Fourier sum $P_N u$ are caused by the slow convergence of

$$F_N(x,\alpha,[u]) = \sum_{k=-N}^{N}\hat{f}_k(\alpha,[u])e^{ikx} \tag{5.6.7}$$

to the sawtooth function $F(x,\alpha,[u])$. Therefore, those oscillations can be eliminated by adding a sawtooth function to the basis of the space to which $u(x)$ is projected. To be specific, we seek an expansion of the form

$$v_N(x) = \sum_{|k|\leqslant N} a_k e^{ikx} + \sum_{|k|>N}\frac{A}{ik}e^{-iky}e^{ikx} \tag{5.6.8}$$

to approximate $u(x)$. The $2N+3$ unknowns a_k , ($|k| \leqslant N$), A and y are determined by the orthogonality condition

$$\int_0^{2\pi} (u - v_N) e^{-ijx} \mathrm{d}x = 0\,, \qquad |j| \leqslant N+2. \tag{5.6.9}$$

The system of equations (5.6.9) leads to the conditions

$$a_k = \hat{u}_k\,, \qquad |k| \leqslant N, \tag{5.6.10}$$

$$\frac{A}{i(N+j)} e^{-i(N+j)y} = \hat{u}_{N+j}\,, \quad j = 1, 2, \tag{5.6.11}$$

where $\hat{u}_k$ are the usual Fourier coefficients of $u(x)$, defined by (5.5.2). Solving equations (5.6.11) gives

$$e^{iy} = \frac{(N+1)\hat{u}_{N+1}}{(N+2)\hat{u}_{N+2}}\,, \qquad A = i(N+1) e^{i(N+1)y} \hat{u}_{N+1}\,. \tag{5.6.12}$$

The procedure described in (5.6.12) is second-order accurate in the location and jump of the shock. In fact, we can state

Theorem 5.1 *Let $u(x)$ be a piecewise C^∞ function with one discontinuity at $x = \alpha$. Let y and A be defined in* (5.6.12)*. Then*

$$|y - \alpha| = \mathcal{O}(N^{-2}), \qquad |A - [u]| = \mathcal{O}(N^{-2}). \tag{5.6.13}$$

Proof It follows from (5.6.3) that

$$\begin{aligned} e^{iy} = \frac{(N+1)\hat{u}_{N+1}}{(N+2)\hat{u}_{N+2}} &= \frac{e^{-i(N+1)x_s}\left[\,[u] + \dfrac{[u']}{i(N+1)} + \mathcal{O}\left(\dfrac{1}{(N+1)^2}\right)\right]}{e^{-i(N+2)x_s}\left[\,[u] + \dfrac{[u']}{i(N+2)} + \mathcal{O}\left(\dfrac{1}{(N+2)^2}\right)\right]} \\ &= e^{ix_s}\left[\,1 + \mathcal{O}(N^{-2})\,\right]. \end{aligned}$$

By the same token,

$$\begin{aligned} |A| &= (N+1)|\hat{u}_{N+1}| = \left[\left\{[u] - \frac{[u'']}{(N+1)^2}\right\}^2 + \frac{[u']^2}{(N+1)^2}\right]^{1/2} \\ &= |[u]|\left[\,1 + \mathcal{O}(N^{-2})\,\right]. \end{aligned}$$

This completes the proof of Theorem 5.1. □

Essentially non-oscillatory reconstruction

Formally, we obtain from (5.6.5), (5.6.8) and (5.6.10) that

$$v_N(x) = \hat{u}_0 - A(\pi - y) + \sum_{\substack{k=-N \\ k\neq 0}}^{N} \left(\hat{u}_k - \frac{A}{ik} e^{-iky} \right) e^{ikx} + F(x, y, A), \quad (5.6.14)$$

where the function F is defined by (5.6.4). Applying appropriate filters, we modify (5.6.14) to give a formula for computing the approximation to u:

$$v_N(x) = \hat{u}_0 - A(\pi - y) + \sum_{\substack{k=-N \\ k\neq 0}}^{N} \sigma(2\pi k/N) \left(\hat{u}_k - \frac{A}{ik} e^{-iky} \right) e^{ikx} + F(x, y, A)\,. \quad (5.6.15)$$

Note that (5.6.12) is an asymptotic formula for the jump location y and strength A. In practice, it is found that the coefficients of modes in the range $(\sqrt{N}, N^{0.75})$ give the best results to detect shock location and strength. Therefore, we choose $\sqrt{N} < N_1 < N^{0.75}$ and solve A and y by the following formulas:

$$e^{iy} = \frac{(N_1+1)\hat{u}_{N_1+1}}{(N_1+2)\hat{u}_{N_1+2}}\,, \qquad A = i(N_1+1)e^{i(N_1+1)y}\hat{u}_{N_1+1}\,. \quad (5.6.16)$$

A pseudocode outlines the above procedure is provided below:

```
CODE Shock.2
Input N, N1, u0(x)
   Δx=2π/(2N+1), xj=jΔx   0⩽j⩽2N
Compute Fourier coefficients ûk, for |k|⩽N
% Compute the jump position y
   y = -i*log((N1+1)*û_{N1+1}/(N1+2)*û_{N1+2})
   y=Re(y)
% Compute the strength of the jump
   A=i*(N1+1)*exp(i(N1+1)y)*û_{N1+1}
   A=Re(A)
% To recover the pointwise value from the
Fourier coefficients
For j=0 to 2N do
% Compute the last term in (5.6.15)
         if xj⩽y then F=-A*xj
```

```
          else F=A*(2π-x_j)
          endif
% Compute the approximations
   u(x_j)=û_0-A*(π-y)+σ(2πk/N)*(û_k-A/(ik)*exp(-i*k*y))
        *exp(i*k*x_j)+F
endFor
```

Example 5.6.1 *We use the above pseudocode on the following function*

$$u(x) = \begin{cases} \sin(x/2), & 0 \leqslant x \leqslant 1.9, \\ -\sin(x/2), & 1.9 < x < 2\pi. \end{cases} \tag{5.6.17}$$

Notice that $[u^{(k)}] \neq 0$ for all $k \geqslant 0$. Using (5.5.2) we obtain for all $k \geqslant 0$,

$$\hat{u}_k = \frac{1}{2\pi}\left(-\frac{e^{-i(k+0.5)*1.9}}{k+0.5} + \frac{e^{-i*(k-0.5)*1.9}}{k-0.5}\right). \tag{5.6.18}$$

Numerical results using `CODE Shock.2` are plotted in Figures 5.9 and 5.10. In the following table, we list the errors of the jump location and its strength determined by `CODE Shock.2`. The filter function used in the code is

$$\sigma(\eta) = e^{-15\ln 10\eta^{12}}.$$

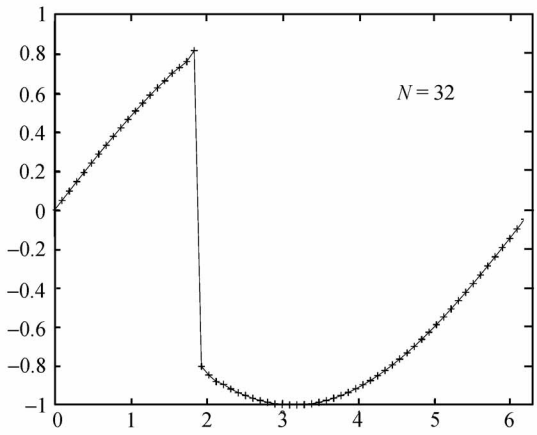

Figure 5.9 The solid line is the exact solution and the pulses the numerical solution with $N = 32$.

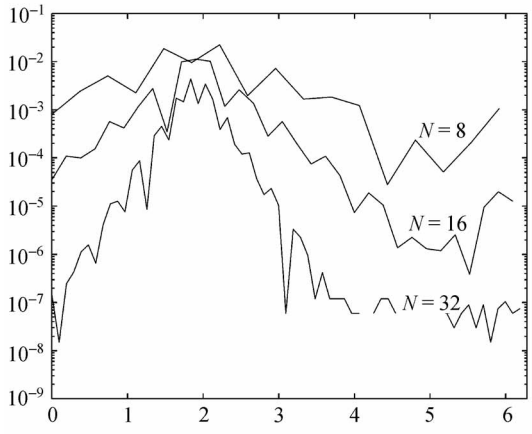

Figure 5.10 Error of the re-construction on logarithm scale for $N = 8, 16, 32$.

Notice that the second-order accuracy is verified.

	Location (exact:1.9)		Strength (exact:-sin(0.95)/π)	
N	error	order	error	order
8	1.1e-02		3.4e-04	
16	3.4e-03	1.69	9.8e-05	1.79
32	9.2e-04	1.89	2.6e-05	1.91
64	2.4e-04	1.94	6.8e-06	1.93

In obtaining the convergence order, we have used the formula:

$$\text{order} = \log_2 \left(\frac{\texttt{error}(h)}{\texttt{error}(h/2)} \right).$$

In using the filters, we choose the parameters $\alpha = m = 4, k_0 = 0$.

We remark that if u is smooth, (5.6.8) keeps spectral accuracy because A determined by (5.6.12) will be spectrally small.

We now state our scheme as (5.5.13) with

$$\hat{f}_{j+1/2} = f(v_N(x_{j+1/2}, t)), \tag{5.6.19}$$

where v_N is defined by (5.6.8). We obtain the Fourier coefficients $\bar{a}_k$ of $\bar{u}$ from $\{\bar{u}_j\}$ by collocation, and obtain a_k of u needed in (5.6.8) by (5.5.16). The main difference between the conventional spectral method and the current approach is that we use the *essentially* non-oscillatory reconstruction v_N instead of the oscillatory $P_N u$ in (5.5.17).

To discretize (5.5.13) in time, we use the high-order TVD Runge-Kutta methods (5.5.18). A pseudocode outlines the above procedure is provided below:

```
CODE Shock.3
Input N, α_jk and β_jk, 1⩽j⩽3, 0⩽k⩽2
Input Δt, u_0(x), T
      Δx=2π/(2N+1), x_j=jΔx  0⩽j⩽2N
Compute ū(j,0)=ū_0(x_j), |j|⩽2N,
using (5.5.11)
time=0
While time⩽T
For r=0 to 2 do
  % Compute Fourier coefficients of ā_k and a_k
  using collocation method
    for |k|⩽ N do
      ā_k= (∑_{j=0}^{2N} ū(j,r) e^{-ikx_j})/(2N+1); τ_k=sin(kΔx/2)/(kΔx/2);
      a_k=ā_k/τ_k
    endfor
  %Compute the jump position y
     N1=N^0.6; y = -i*log((N1+1)*a_{N1+1}/(N1+2)*a_{N1+2}); y=Re(y)
  %Compute the strength of the jump
     A=i*(N1+1)*exp(i(N1+1)y)*a_{N1+1}; A=Re(A)
  %To recover pointwise value from Fourier coefficients
   for j=0 to 2N do
  %Compute the last term in (5.6.15)
      if x_j⩽y then F=-A*x_j
        else F=A*(2π-x_j)
      endif
  %Compute u(x_{j+1/2}) using a weak filter
      x_{j+1/2}=x_j+0.5*Δx
      u(x_{j+1/2})=a_0-A(π-y)+∑_{k≠0} σ(2πk/N)(a_k-Ae^{-iky}/(ik)) e^{ikx_{j+1/2}}+F
   endfor
  %Runge-Kutta method
    for j=0 to 2N do
       RHS(j,r)=-(f(u(x_{j+1/2}))-f(u(x_{j-1/2})))/Δx
       if r=0 then ū(j,1)=ū(j,0)+Δt RHS(j,0)
        elseif r=1 then ū(j,2)=3/4 ū(j,0)+1/4 ū(j,1)+Δt/4 RHS(j,1)
        elseif r=2 then ū(j,3)=1/3 ū(j,0)+2/3 ū(j,2)+2/3 ΔtRHS(j,2)
       endif
endFor
  Update the initial value:  ū(j,0)=ū(j,3), 0⩽j⩽2N
  time=time+Δt
endWhile
```

```
%Final solution smoothing using a stronger filter σ̃
    for |k|⩽ N do
        ā_k=(Σ_{j=0}^{2N} ū_j e^{-ikx_j})/(2N+1); a_k=ā_k/τ_k
    endfor
    for j=0 to 2N do
       u(x_j)=a_0-A(π-y)+Σ_{k≠0} σ̃(2πk/N)(a_k-Ae^{-iky}/(ik))e^{ikx_j}+F
    endfor
```

We now reconsider Example 5.6.1 by using `CODE Shock.3`. The weak filter σ used above is $\sigma(\eta) = e^{-15(\ln 10)\eta^8}$, and a strong filter $\tilde{\sigma}$ used is $\sigma(\eta) = e^{-15(\ln 10)\eta^4}$.

The numerical solution with $T = 0.5$ and $N = 32$ is displayed in Figure 5.11. At $t = 2$, we employed a coarse grid with $N = 32$ and a finer grid with $N = 64$. The convergence with respect to the mesh size is observed from Figures 5.12 and 5.13.

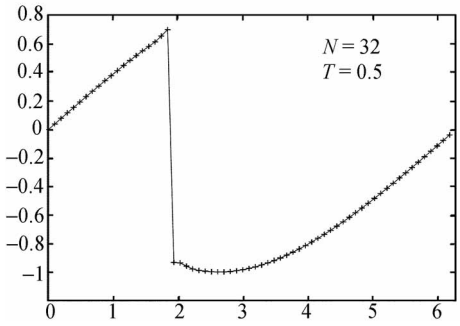

Figure 5.11 Inviscid Burgers' equation with the initial function (5.6.17) ($N = 32$ and $t = 0.5$).

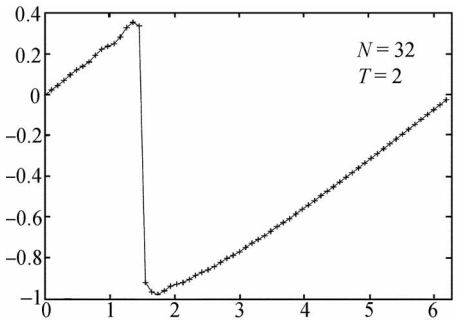

Figure 5.12 Same as Figure 5.11, except $t = 2$.

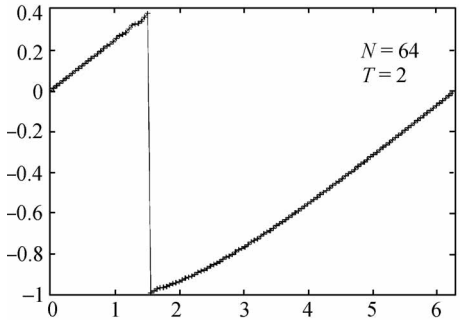

Figure 5.13 Same as Figure 5.12, except $N = 64$.

Chapter 6

Spectral methods in Multi-dimensional Domains

Contents

6.1 Spectral-collocation methods in rectangular domains 233

6.2 Spectral-Galerkin methods in rectangular domains 237

6.3 Spectral-Galerkin methods in cylindrical domains 243

6.4 A fast Poisson Solver using finite differences 247

In this chapter, we are mainly concerned with spectral approximations for the following model problem:

$$\alpha u - \Delta u = f \tag{6.0.1}$$

in a regular domain Ω with appropriate boundary conditions.

Developing efficient and accurate numerical schemes for (6.0.1) is very important since

- (i) one often needs to solve (6.0.1) repeatedly after a semi-implicit time discretization of many parabolic type equations;
- (ii) as in the one-dimensional case, it can be used as a preconditioner for more

general second-order problems with variable coefficients, such as

$$Lu := -\sum_{i,j=1}^{d} D_i(a_{ij}D_j u) + \sum_{i=1}^{d} D_i(b_i u) + hu = f,\ \boldsymbol{x} \in \Omega. \tag{6.0.2}$$

Unlike in the one-dimensional case, it is generally not feasible to solve the (non-separable) equation (6.0.2) directly using a spectral method. In other words, for (6.0.2) with variable coefficients, it is necessary to use a preconditioned iterative method.

Computational costs for multidimensional problems using spectral methods could easily become prohibitive if the algorithms are not well designed. There are two key ingredients which make spectral methods feasible for multidimensional problems.

The first is the classical method of "separation of variables" which write the solution of a multidimensional separable equation as a product of functions with one independent variable. We shall explore this approach repeatedly in this chapter.

The second is the observation that spectral transforms in multidimensional domains can be performed through *partial summation*. For example, Orszag[125] pointed out that one can save a factor of 10,000 in the computer time for his turbulence code CENTICUBE ($128 \times 128 \times 128$ degrees of freedom) merely by evaluating the multidimensional spectral transforms through *partial summation*. We will illustrate his idea by a two-dimensional example.

Suppose the goal is to evaluate an $M \times N$ spectral sum at each point of the interpolating grid. Let the sum be

$$f(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} a_{mn}\phi_m(x)\phi_n(y). \tag{6.0.3}$$

To compute (6.0.3) at an *arbitrary* point as a *double* DO LOOP, a total of MN multiplications and MN additions are needed even if the values of the basis functions have been computed and stored.

Since there are MN points on the collocation grid, we would seem to require a total $\mathcal{O}(M^2N^2)$ operations to perform a two-dimensional transform from series coefficients to grid point values. Thus, if M and N are the same order of magnitude, the operation count for each such transform increases as the fourth power of the number of degrees in x direction – and we have to do this once per time step. A finite difference method, in contrast, requires only $\mathcal{O}(MN)$ operations per time step.

Now arrange (6.0.3) as

$$f(x,y)=\sum_{m=0}^{M-1}\phi_m(x)\Big[\sum_{n=0}^{N-1}a_{mn}\phi_n(y)\Big]. \tag{6.0.4}$$

Let us define the *line functions* via $f_j(x)=f(x,y_j), 0\leqslant j\leqslant N-1$. It follows from (6.0.4) that

$$f_j(x)=\sum_{m=0}^{M-1}\alpha_m^{(j)}\phi_m(x),\quad 0\leqslant j\leqslant N-1,$$

where

$$\alpha_m^{(j)}=\sum_{n=0}^{N-1}a_{mn}\phi_n(y_j),\quad 0\leqslant m\leqslant M-1,\ 0\leqslant j\leqslant N-1. \tag{6.0.5}$$

There are MN coefficients $\alpha_m^{(j)}$, and each is a sum over N terms as in (6.0.5), so the expense of computing the spectral coefficients of the $f_j(x)$ is $\mathcal{O}(MN^2)$. Each $f_j(x)$ describes how $f(x,y)$ varies with respect to x on a particular grid line, so we can evaluate $f(x,y)$ everywhere on the grid. Since $f_j(x)$ are *one-dimensional*, each can be evaluated at a single point in only $\mathcal{O}(M)$ operations:

Conclusion:

• In two-dimensions, [direct sum] $\mathcal{O}(M^2N^2)\to\mathcal{O}(MN^2)+\mathcal{O}(M^2N)$ [partial sum];

• In three-dimensions, $L\times M\times N$ points on x,y and z directions: [direct sum] $\mathcal{O}(L^2M^2N^2)\to\mathcal{O}(LMN^2)+\mathcal{O}(LM^2N)+\mathcal{O}(L^2MN)$ [partial sum];

• The cost of partial summation can be reduced further to $\mathcal{O}(NM\log(NM))$ in two-dimensions and $\mathcal{O}(LMN\log(LMN))$ in three-dimensions if we are dealing with a Fourier or Chebyshev expansion.

In the rest of this chapter, we shall present several efficient numerical algorithms for solving (6.0.1).

6.1 Spectral-collocation methods in rectangular domains

Let $\Omega=(-1,1)^2$. We consider the two-dimensional Poisson type equation

$$\begin{cases}\alpha u-\Delta u=f, & \text{in }\Omega,\\ u(x,y)=0, & \text{on }\partial\Omega.\end{cases} \tag{6.1.1}$$

For the sake of simplicity, we shall use the same number of points, N, in the x and y directions, although in practical applications one may wish to use different numbers of points in each direction. Let $\mathbf{X_N} = \{u \in P_N \times P_N : \ u|_{\partial\Omega} = 0\}$ and $\{\xi_i\}_{i=0}^N$ be the Chebyshev or Legendre Gauss-Lobatto points. Then, the Chebyshev- or Legendre-collocation method is to look for $u_N \in \mathbf{X}_N$ such that

$$\alpha u_N(\xi_i,\xi_j) - \partial_x^2 u_N(\xi_i,\xi_j) - \partial_y^2 u_N(\xi_i,\xi_j) = f(\xi_i,\xi_j),\ 1 \leqslant i,j \leqslant N-1. \quad (6.1.2)$$

Let $\{h_n(\xi)\}_{n=0}^N$ be the Lagrange polynomial associated with $\{\xi_i\}_{i=0}^N$. We can write

$$u_N(x,y) = \sum_{m=0}^{N}\sum_{n=0}^{N} u_N(\xi_m,\xi_n)h_m(x)h_n(y).$$

Let D_2 be the second-order differentiation matrix, given in Section 2.4 for the Chebyshev case, and let U and F be two matrices of order $(N-1)\times(N-1)$ such that

$$U = (u_N(\xi_m,\xi_n))_{m,n=1}^{N-1}, \quad F = (f(\xi_m,\xi_n))_{m,n=1}^{N-1}.$$

Then, (6.1.2) becomes the *matrix* equation

$$\alpha U - D_2 U - U D_2^{\mathrm{T}} = F, \quad (6.1.3)$$

which can also be written as a standard linear system,

$$(\alpha I \otimes I - I \otimes D_2 - D_2 \otimes I)\bar{u} = \bar{f}, \quad (6.1.4)$$

where I is the identity matrix, $\bar{f}$ and $\bar{u}$ are vectors of length $(N-1)^2$ formed by the columns of F and U, and $\otimes$ denotes the tensor product of matrices, i.e. $A \otimes B = (Ab_{ij})_{i,j=1}^{N-1}$.

Since D_2 is a full matrix, a naive approach using Gauss elimination for (6.1.4) would cost $\mathcal{O}(N^6)$ operations. However, this cost can be significantly reduced by using a discrete version of "separation of variables" — the matrix decomposition method[112], known as the matrix diagonalization method in the field of spectral methods [79, 80]. To this end, we consider the eigenvalue problem

$$D_2\bar{x} = \lambda\bar{x}. \quad (6.1.5)$$

It has been shown (cf. [58]) that the eigenvalues of D_2 are all negative and distinct. Hence, it is diagonalizable, i.e., if Λ is the diagonal matrix whose diagonal entries $\{\lambda_p\}$ are the eigenvalues of (6.1.5), and let P be the matrix whose columns are the

eigenvectors of (6.1.5), then we have

$$P^{-1}D_2P = \Lambda, \tag{6.1.6}$$

Multiplying (6.1.3) from the left by P^{-1} and from the right by P^{-T}, we find that

$$\begin{aligned} &\alpha P^{-1}U(P^{-1})^{\mathrm{T}} - (P^{-1}D_2P)(P^{-1}U(P^{-1})^{\mathrm{T}}) \\ &- (P^{-1}U(P^{-1})^{\mathrm{T}})(P^{\mathrm{T}}D_2^{\mathrm{T}}(P^{-1})^{\mathrm{T}}) = P^{-1}F(P^{-1})^{\mathrm{T}}. \end{aligned} \tag{6.1.7}$$

Let $\tilde{U} = P^{-1}U(P^{-1})^{\mathrm{T}}$ and $\tilde{F} = P^{-1}F(P^{-1})^{\mathrm{T}}$. Then (6.1.7) becomes

$$\alpha\tilde{U} - \Lambda\tilde{U} - \tilde{U}\Lambda^{\mathrm{T}} = \tilde{F},$$

which gives

$$\tilde{U}_{i,j} = \frac{\tilde{F}_{i,j}}{\alpha - \Lambda_{ii} - \Lambda_{jj}}, \qquad 1 \leqslant i, j \leqslant N-1. \tag{6.1.8}$$

Solving the above equations gives the matrix $\tilde{U}$. Using the relation $U = P\tilde{U}P^{\mathrm{T}}$ yields the solution matrix U.

In summary, the solution of (6.1.2) consists of the following steps:

- Step 1: Pre-processing: compute the eigenvalues and eigenvectors (Λ, P) of D_2;
- Step 2: Compute $\tilde{F} = P^{-1}F(P^{-1})^{\mathrm{T}}$;
- Step 3: Compute $\tilde{U}$ from (6.1.8);
- Step 4: Obtain the solution $U = P\tilde{U}P^{\mathrm{T}}$.

We note that the main cost of this algorithm is the four matrix-matrix multiplications in Steps 2 and 4. Hence, besides the cost of pre-computation, the cost for solving each equation is about $4N^3$ flops, no matter whether Chebyshev or Legendre points are used. We also note that the above algorithms can be easily extended to three-dimensional cases, we refer to [80].

Example 6.1.1 *Solve the 2D Poisson equation*

$$\begin{aligned} &u_{xx} + u_{yy} = 10\sin(8x(y-1)), \quad (x,y) \in \Omega, \\ &u(x,y)|_{\partial\Omega} = 0, \end{aligned} \tag{6.1.9}$$

with the Chebyshev-collocation method.

A simple, but not very efficient, MATLAB code which solves (6.1.4) directly is provided below. The code begins with the differentiation matrix and the Chebyshev-Gauss-Lobatto points, which was described in detail in Chapters 1 & 2.

```
  CODE Poisson.m
% Solve Poisson eqn on [-1,1]x[-1,1] with u=0 on boundary
%D= differentiation matrix -- from DM.4 in Sect.  2.1
%Input N
   x = cos(pi*(0:N)/N)'; y = x;
% Set up grids and tensor product Laplacian, and solve for u:
  [xx,yy] = meshgrid(x(2:N),y(2:N));
% stretch 2D grids to 1D vectors
  xx = xx(:); yy = yy(:);
% source term function
  f = 10*sin(8*xx.*(yy-1));
  D2 = D^2; D2 = D2(2:N,2:N); I = eye(N-1);
% Laplacian
  L = kron(I,D2) + kron(D2,I);
  figure(1), clf, spy(L), drawnow
%solve problem and watch clock
  tic, u = L\f; toc
% Reshape long 1D results onto 2D grid:
  uu = zeros(N+1,N+1); uu(2:N,2:N) = reshape(u,N-1,N-1);
  [xx,yy] = meshgrid(x,y);
  value = uu(N/4+1,N/4+1);
% Interpolate to finer grid and plot:
  [xxx,yyy] = meshgrid(-1:.04:1,-1:.04:1);
  uuu = interp2(xx,yy,uu,xxx,yyy,'cubic');
  figure(2), clf, mesh(xxx,yyy,uuu), colormap(1e-6*[1 1 1]);
  xlabel x, ylabel y, zlabel u
  text(.4,-.3,-.3,sprintf('u(2^-1/2,2^-1/2)=%14.11f',value))
```

Exercises 6.1

Problem 1 Solve the Poisson problem

$$\begin{aligned} &u_{xx} + u_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y), \qquad (x,y) \in \Omega = (-1,1)^2, \\ &u(x,y)|_{\partial\Omega} = 0. \end{aligned} \tag{6.1.10}$$

using the Chebyshev pseudo-spectral method with formula (6.1.8). The exact solution of this problem is $u(x,y) = \sin(\pi x)\sin(\pi y)$.

Problem 2 Consider the Poisson problem

$$\begin{aligned} &u_{xx}+u_{yy}+au_x+bu_y=f(x,y), \qquad (x,y)\in\Omega=(-1,1)^2,\\ &u(x,y)|_{\partial\Omega}=0, \end{aligned} \tag{6.1.11}$$

where a and b are constants.

a. Derive a Chebyshev pseudo-spectral method for solving this problem.

b. Let $a=b=1$ and $f(x,y)=-2\pi^2\sin(\pi x)\sin(\pi y)+\pi(\cos(\pi x)\sin(\pi y)+\sin(\pi x)\cos(\pi y))$.

The exact solution of this problem is $u(x,y)=\sin(\pi x)\sin(\pi y)$. Solve the problem using your code in part (a).

Problem 3 Consider the following two dimensional *separable* equation in $\Omega=(-1,1)^2$:

$$\begin{aligned} &a(x)u_{xx}+b(x)u_x+c(x)u+d(y)u_{yy}+e(y)u_y+f(y)u=g(x,y),\\ &u|_{\partial\Omega}=0. \end{aligned} \tag{6.1.12}$$

Design an efficient spectral-collocation method for solving this equation.

Problem 4 Write down the matrix diagonalization algorithm for the Poisson type equation in $\Omega=(-1,1)^3$.

6.2 Spectral-Galerkin methods in rectangular domains

Matrix diagonalization method
Legendre case
Chebyshev case
Neumann boundary conditions

The weighted spectral-Galerkin approximation to (6.1.1) is: Find $u_N\in \boldsymbol{X}_N$ such that

$$\alpha(u_N,v_N)_\omega+a_\omega(u_N,v_N)=(I_Nf,v_N)_\omega \ \text{ for all } \ v_N\in\boldsymbol{X}_N, \tag{6.2.1}$$

where $I_N: C(\Omega)\longrightarrow P_N^d$ is the interpolation operator based on the Legendre or Chebyshev Gauss-Lobatto points, $(u,v)_\omega=\int_\Omega uv\omega\mathrm{d}\boldsymbol{x}$ is the inner product in $L^2_\omega(\Omega)$ and

$$a_\omega(u,v)=(\nabla u,\omega^{-1}\nabla(v\omega))_\omega. \tag{6.2.2}$$

Matrix diagonalization method

Let $\{\phi_k\}_{k=0}^{N-2}$ be a set of basis functions for $P_N \cap H_0^1(I)$. Then,

$$\boldsymbol{X}_N = \text{span}\{\phi_i(x)\phi_j(y) : i,j = 0,1,\cdots,N-2\}.$$

Let us denote

$$\begin{aligned}
&u_N = \sum_{k,j=0}^{N-2} \tilde{u}_{kj}\phi_k(x)\phi_j(y), \quad f_{kj} = (I_N f, \phi_k(x)\phi_j(y))_\omega,\\
&s_{kj} = \int_I \phi_j'(x)(\phi_k(x)\omega(x))'\mathrm{d}x, \quad S = (s_{kj})_{k,j=0,1,\cdots,N-2},\\
&m_{kj} = \int_I \phi_j(x)\phi_k(x)\omega(x)\mathrm{d}x, \quad M = (m_{kj})_{k,j=0,1,\cdots,N-2},\\
&U = (\tilde{u}_{kj})_{k,j=0,1,\cdots,N-2}, \ F = (f_{kj})_{k,j=0,1,\cdots,N-2}.
\end{aligned} \tag{6.2.3}$$

Taking $v_N = \phi_l(x)\phi_m(y)$ in (6.2.1) for $l,m = 0,1,\cdots,N-2$, we find that (6.2.1) is equivalent to the matrix equation

$$\alpha MUM + SUM + MUS^{\mathrm{T}} = F. \tag{6.2.4}$$

We can also rewrite the above matrix equation in the following form using the tensor product notation:

$$(\alpha M \otimes M + S \otimes M + M \otimes S^{\mathrm{T}})\bar{u} = \bar{f}, \tag{6.2.5}$$

where, as in the last section, $\bar{f}$ and $\bar{u}$ are vectors of length $(N-1)^2$ formed by the columns of U and F. As in the spectral collocation case, this equation can be solved in particular by the matrix diagonalization method. To this end, we consider the generalized eigenvalue problem:

$$M\bar{x} = \lambda S\bar{x}. \tag{6.2.6}$$

In the Legendre case, M and S are symmetric positive definite matrices so all the eigenvalues are real positive. In the Chebyshev case, S is no longer symmetric but it is still positive definite. Furthermore, it is shown in [58] that all the eigenvalues are real, positive and distinct. Let Λ be the diagonal matrix whose diagonal entries $\{\lambda_p\}$ are the eigenvalues of (6.2.6), and let E be the matrix whose columns are the eigenvectors of (6.2.6). Then, we have

$$ME = SE\Lambda. \tag{6.2.7}$$

Now setting $U = EV$, thanks to (6.2.7) the equation (6.2.4) becomes

$$\alpha SE\Lambda VM + SEVM + SE\Lambda VS^{\mathrm{T}} = F. \tag{6.2.8}$$

Multiplying $E^{-1}S^{-1}$ to the above equation, we arrive at

$$\alpha\Lambda VM + VM + \Lambda VS^{\mathrm{T}} = E^{-1}S^{-1}F := G. \tag{6.2.9}$$

The transpose of the above equation reads

$$\alpha MV^{\mathrm{T}}\Lambda + MV^{\mathrm{T}} + SV^{\mathrm{T}}\Lambda = G^{\mathrm{T}}. \tag{6.2.10}$$

Let $\bar{v}_p = (v_{p0}, v_{p1}, \cdots, v_{pN-2})^{\mathrm{T}}$ and $\bar{g}_p = (g_{p0}, g_{p1}, \cdots, g_{pN-2})^{\mathrm{T}}$ for $0 \leqslant p \leqslant N-2$. Then the p-th column of equation (6.2.10) can be written as

$$((\alpha\lambda_p + 1)M + \lambda_p S)\,\bar{v}_p = \bar{g}_p, \quad p = 0, 1, \cdots, N-2. \tag{6.2.11}$$

These are just the $N-1$ linear systems from the Legendre- or Chebyshev-Galerkin approximation of the $N-1$ one-dimensional equations

$$(\alpha\lambda_p + 1)v_p - \lambda_p v_p'' = g_p, \quad v_p(\pm 1) = 0.$$

Note that we only diagonalize in the x-direction and reduce the problem to $N-1$ one-dimensional equations (in the y-direction) (6.2.11) for which, unlike in the collocation case, a fast algorithm is available.

In summary, the solution of (6.2.4) consists of the following steps:

- **Step 1** Pre-processing: compute the eigenvalues and eigenvectors of the generalized eigenvalue problem (6.2.6);
- **Step 2** Compute the expansion coefficients of $I_N f$ (backward Legendre or Chebyshev transform);
- **Step 3** Compute $F = (f_{ij})$ with $f_{ij} = (I_N f, \phi_i(x)\phi_j(y))_\omega$;
- **Step 4** Compute $G = E^{-1}S^{-1}F$;
- **Step 5** Obtain V by solving (6.2.11);
- **Step 6** Set $U = EV$;
- **Step 7** Compute the values of u_N at Gauss-Lobatto points (forward Legendre or Chebyshev transform).

Several remarks are in order:

Remark 6.2.1 This algorithm is slightly more complicated than the spectral-collocation algorithm presented in the last section but it offers several distinct advantages:

• Unlike in the collocation case, the eigenvalue problems here involve only sparse (or specially structured) matrices so it can be computed much more efficiently and accurately.

• This algorithm can be easily applied to problems with general boundary conditions (3.2.2) since we only have to modify the basis functions and the associated stiffness and mass matrices.

• For the Dirichlet boundary conditions considered here, the basis functions take the form $\phi_k(x) = a_k p_k(x) + b_k p_{k+2}(x)$ where $p_k(x)$ is either the Legendre or Chebyshev polynomial. Thanks to the odd-even parity of the Legendre or Chebyshev polynomials, the matrices S and M can be split into two sub-matrices of order $N/2$ and $N/2 - 1$. Consequently, (6.2.4) can be split into four sub-equations. Hence, the cost of matrix-matrix multiplications in the above procedure can be cut by half. The above remark applies also to the Neumann boundary conditions but not to the general boundary conditions.

• The main cost of this algorithm is the two matrix multiplications in Steps 4 & 6 plus the backward and forward transforms. However, it offers both the nodal values of the approximate solution as well as its expansion coefficients which can be used to compute its derivatives, a necessary step in any real application code, at negligible cost. Hence, this algorithm is also efficient.

• The above procedure corresponds to diagonalizing in the x direction; one may of course choose to diagonalize in the y direction. In fact, if different numbers of modes are used in each direction, one should choose to diagonalize in the direction with fewer modes to minimize the operational counts of the two matrix-matrix multiplications in the solution procedure.

Legendre case

Let $\phi_k(x) = (L_k(x) - L_{k+2}(x))/\sqrt{4k+6}$. Then, we have $S = I$ and M can be split into two symmetric tridiagonal sub-matrices so the eigenvalues and eigenvectors of M can be easily computed in $\mathcal{O}(N^2)$ operations by standard procedures. Furthermore, we have $E^{-1} = E^{\mathrm{T}}$. Step 2 consists of solving $N - 1$ tridiagonal systems of order $N - 1$. Therefore, for each right-hand side, the cost of solving system (6.2.4) is dominated by the two matrix-matrix multiplications in Steps 4 $\&$ 6.

Chebyshev case:

Let $\phi_k(x) = T_k(x) - T_{k+2}(x)$. Then, S is a special upper triangular matrix given in (3.3.8) and M is a symmetric positive definite matrix with three non-zero diagonals. Similar to the Legendre case, S and M can be split into two sub-matrices so that the eigen-problem (6.2.6) can be split into four subproblems which can be solved directly by using a QR method. Note that an interesting $\mathcal{O}(N^2)$ algorithm for solving (6.2.6) was developed in [15]. Once again, the cost of solving system (6.2.4) in the Chebyshev case is also dominated by the two matrix-matrix multiplications in Steps 4 & 6.

Neumann boundary conditions

The matrix diagonalization approach applies directly to separable elliptic equations with general boundary conditions including in particular the Neumann boundary conditions. However, the problem with Neumann boundary conditions

$$\begin{aligned} &\alpha u - \Delta u = f \qquad \text{in } \Omega; \\ &\frac{\partial u}{\partial n}|_{\partial\Omega} = 0 \end{aligned} \tag{6.2.12}$$

needs some special care, especially when $\alpha = 0$ since the solution u of (6.2.12) is only determined up to an additive constant.

Since one often needs to deal with the problem (6.2.12) in practice, particularly in a projection method for solving time-dependent Navier-Stokes equations (cf. Section 7.4), we now describe how the matrix diagonalization method needs to be modified for (6.2.12). In this case, we have from Remark 3.2.1 that in the Legendre case,

$$\phi_k(x) = L_k(x) - \frac{k(k+1)}{(k+2)(k+3)} L_{k+2}(x), \quad k = 1, \cdots, N-2, \tag{6.2.13}$$

and from Remark 3.3.9 that in the Chebyshev case,

$$\phi_k(x) = T_k(x) - \frac{k^2}{(k+2)^2} T_{k+2}(x), \quad k = 1, \cdots, N-2, \tag{6.2.14}$$

For multidimensional problems, this set of basis functions should be augmented with $\phi_0(x) = 1/\sqrt{2}$ in the Legendre case and $\phi_0(x) = 1/\sqrt{\pi}$ in the Chebyshev case.

For $\alpha > 0$, we look for an approximate solution in the space

$$X_N = \text{span}\{\phi_i(x)\phi_j(y) : \ 0 \leqslant i, j \leqslant N-2\}. \tag{6.2.15}$$

However, for $\alpha = 0$, where the solution u of (6.2.12) is only determined up to an additive constant, we fix this constant by setting $\int_\Omega u\omega(x)\omega(y)\mathrm{d}x\mathrm{d}y = 0$, assuming that the function f satisfies the compatibility condition $\int_\Omega f\omega(x)\omega(y)\mathrm{d}x\mathrm{d}y = 0$. In this case, we set

$$X_N = \text{span}\{\phi_i(x)\phi_j(y) :\ 0 \leqslant i, j \leqslant N-2; i \text{ or } j \neq 0\}. \tag{6.2.16}$$

Using the same notations in (6.2.3), we find that the Legendre-Galerkin approximation to (6.2.12) can still be written as the matrix equation (6.2.4) with

$$M = \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & M_1 \end{pmatrix}, \qquad S = \begin{pmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & S_1 \end{pmatrix}, \tag{6.2.17}$$

where M_1 and S_1 are the mass and stiffness matrices of order $N-2$ corresponding to $\{\phi_k\}_{k=1}^{N-2}$.

Let us now consider the generalized eigenvalue problem $M_1\bar{x} = \lambda S_1\bar{x}$, and let (Λ_1, E_1) be such that

$$M_1E_1 = S_1E_1\Lambda_1. \tag{6.2.18}$$

Setting

$$E = \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & E_1 \end{pmatrix},\ \Lambda = \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \Lambda_1 \end{pmatrix},\ \tilde{S} = \begin{pmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & S_1 \end{pmatrix},\ \tilde{I} = \begin{pmatrix} 0 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & I_1 \end{pmatrix}, \tag{6.2.19}$$

where I_1 is the identity matrix of order $N-2$, we have

$$ME = \tilde{S}E\Lambda. \tag{6.2.20}$$

Now applying the transform $U = EV$ to (6.2.4), we obtain, thanks to (6.2.20),

$$\alpha\tilde{S}E\Lambda VM + SEVM + \tilde{S}E\Lambda VS^{\mathrm{T}} = F. \tag{6.2.21}$$

Multiplying $E^{-1}\tilde{S}^{-1}$ to the above equation, we arrive at

$$\alpha\Lambda VM + \tilde{I}VM + \Lambda VS^{\mathrm{T}} = E^{-1}\tilde{S}^{-1}F =: G. \tag{6.2.22}$$

The transpose of the above equation reads

$$\alpha MV^{\mathrm{T}}\Lambda + MV^{\mathrm{T}}\tilde{I} + SV^{\mathrm{T}}\Lambda = G^{\mathrm{T}}. \tag{6.2.23}$$

Let $\bar{v}_p = (v_{p0}, v_{p1}, \cdots, v_{pN-2})^{\mathrm{T}}$ and $\bar{g}_p = (g_{p0}, g_{p1}, \cdots, g_{pN-2})^{\mathrm{T}}$ for $0 \leqslant p \leqslant$

$N-2$. Then the p-th column of equation (6.2.23) can be written as

$$((\alpha\lambda_p+1)M+\lambda_p S)\,\bar{v}_p=\bar{g}_p,\quad p=1,\cdots,N-2, \tag{6.2.24}$$

$$(\alpha M+S)\,\bar{v}_0=\bar{g}_0. \tag{6.2.25}$$

Note that for $\alpha=0$, the last equation is only solvable if $g_{00}=0$ (the compatibility condition), and we set $v_{00}=0$ so that we have $\int_\Omega u_N\omega(x)\omega(y)\mathrm{d}x\mathrm{d}y=0$.

Exercise 6.2

Problem 1 Consider the 2D Poisson type equation with the mixed boundary conditions

$$(a_\pm u+b_\pm u_x)(x,\pm1)=0,\quad (c_\pm u+d_\pm u_y)(\pm1,y)=0. \tag{6.2.26}$$

Write down the Legendre-Galerkin method for this problem and design an efficient matrix diagonalization algorithm for it.

Problem 2 Consider the following two dimensional *separable* equation in $\Omega=(-1,1)^2$:

$$\begin{aligned}&a(x)u_{xx}+b(x)u_x+c(x)u+d(y)u_{yy}+e(y)u_y+f(y)u=g(x,y),\\&u(x,\pm1)=0,\ x\in[-1,1];\quad u_x(\pm1,y)=0,\ y\in[-1,1].\end{aligned} \tag{6.2.27}$$

1. Assuming that all the coefficients are constants, design an efficient Legendre-Galerkin algorithm for solving this equation.

2. Assuming that d, e, f are constants, design an efficient Legendre-Galerkin method for solving this equation.

Problem 3 Write down the matrix diagonalization algorithm for the Poisson type equation in $\Omega=(-1,1)^3$ with homogeneous Dirichlet boundary conditions.

6.3 Spectral-Galerkin methods in cylindrical domains

In many practical situations, one often needs to solve partial differential equations in cylindrical geometries. Since a cylinder is a separable domain under the cylindrical coordinates, we can still apply, as in the previous sections, the discrete "separation of variables" to separable equations in a cylinder.

Let us consider for example the Poisson type equation

$$\alpha U-\Delta U=F\quad \text{in }\hat{\Omega};\qquad U|_{\partial\hat{\Omega}}=0, \tag{6.3.1}$$

where $\hat{\Omega} = \{(x, y, z) : x^2 + y^2 < 1, \ -1 < z < 1\}$. Applying the cylindrical transformations $x = r\cos\theta$, $y = r\sin\theta$, $z = z$, and setting $u(r, \theta, z) = U(r\cos\theta, r\sin\theta, z)$, $f(r, \theta, z) = F(r\cos\theta, r\sin\theta, z)$, Eq. (6.3.1) becomes

$$\begin{aligned} &-\frac{1}{r}(ru_r)_r - \frac{1}{r^2}u_{\theta\theta} - u_{zz} + \alpha u = f \quad (r, \theta, z) \in (0, 1) \times [0, 2\pi) \times (-1, 1), \\ &u = 0 \text{ at } r = 1 \text{ or } z = \pm 1, \quad u \text{ periodic in } \theta. \end{aligned} \tag{6.3.2}$$

To simplify the notation, we shall consider the axisymmetric case, i.e., f and u are independent of θ. Note that once we have an algorithm for the axisymmetric case, the full three-dimensional case can be easily handled by using a Fourier method in the θ direction, we refer to [142] for more details in this matter.

Assuming f and u are independent of θ, making a coordinate transformation $r = (t+1)/2$ and denoting $v(t, z) = u(r, z)$, $g(t, z) = (t+1)f(r, z)/4$ and $\beta = \alpha/4$, we obtain a two-dimensional equation

$$\begin{aligned} &-\frac{1}{4}(t+1)v_{zz} - \big((t+1)v_t\big)_t + \beta(t+1)v = g \quad (t, z) \in (-1, 1)^2, \\ &v = 0 \text{ at } t = 1 \text{ or } z = \pm 1. \end{aligned} \tag{6.3.3}$$

Let us denote $\psi_i(z) = p_i(z) - p_{i+2}(z)$ and $\phi_i(t) = p_i(t) - p_{i+1}(t)$, where p_j is either the j-th degree Legendre or Chebyshev polynomial. Let

$$X_N = \text{span}\{\phi_i(t)\psi_j(z) : \ 0 \leqslant i \leqslant N-1, \ 0 \leqslant j \leqslant N-2\}.$$

Then a spectral-Galerkin approximation to (6.3.3) is to find $v_N \in X_N$ such that

$$\begin{aligned} &\frac{1}{4}\big((t+1)\partial_z v_N, \partial_z(w\,\omega)\big) + \big((t+1)\partial_t v_N, \partial_t(w\omega)\big) + \beta\big((t+1)v_N, w\big)_\omega \\ &= (I_{N,\omega}g, w)_\omega, \qquad \text{for all } \ w \in X_N, \end{aligned} \tag{6.3.4}$$

where $\omega \equiv 1$ in the Legendre case and $\omega = \omega(t, z) = \big((1-t^2)(1-z^2)\big)^{-\frac{1}{2}}$ in the Chebyshev case, $(\cdot, \cdot)_\omega$ is the weighted L^2-inner product in $(-1, 1)^2$, and $I_{N,\omega}$ is the interpolation operator based on the Legendre- or Chebyshev-Gauss type points. Setting

$$a_{ij} = \int_I (t+1)\,\phi_j'\,\big(\phi_i\omega(t)\big)'\,\mathrm{d}t, \quad A = (a_{ij})_{0\leqslant i\leqslant N-1, 0\leqslant j\leqslant N-2},$$

$$c_{ij} = \int_I (t+1)\,\phi_j\,\phi_i\,\omega(t)\,\mathrm{d}t, \quad C = (c_{ij})_{0\leqslant i\leqslant N-1, 0\leqslant j\leqslant N-2},$$

$$m_{ij} = \int_I \psi_j \, \psi_i \, \omega(z) \, \mathrm{d}z, \quad M = (m_{ij})_{i,j=0,1,\cdots,N-2},$$

$$s_{ij} = \int_I \psi_j' \left(\psi_i \, \omega(z)\right)' \mathrm{d}z, \quad S = (s_{ij})_{i,j=0,1,\cdots,N-2},$$

and

$$f_{ij} = \int_I \int_I I_{N,\omega} g \, \phi_i(t) \, \psi_j(z) \, \omega(t,z) \, \mathrm{d}t \, \mathrm{d}z, \quad F = (f_{ij})_{0 \leqslant i \leqslant N-1, 0 \leqslant j \leqslant N-2},$$

$$v_N = \sum_{i=0}^{N-1} \sum_{j=0}^{N-2} u_{ij} \phi_i(t) \psi_j(z), \quad U = (u_{ij})_{0 \leqslant i \leqslant N-1, 0 \leqslant j \leqslant N-2}.$$

Then (6.3.4) becomes the matrix equation

$$\frac{1}{4} C U S^t + (A + \beta C) U M = F. \tag{6.3.5}$$

The non-zero entries of M and S are given in (3.2.7) and (3.2.6) in the Legendre case, and in (3.3.7) and (3.3.8) in the Chebyshev case. Using the properties of Legendre and Chebyshev polynomials, it is also easy to determine the non-zero entries of A and C.

In the Legendre case, the matrix A is diagonal with $a_{ii} = 2i + 2$, and the matrix C is symmetric penta-diagonal with

$$c_{ij} = \begin{cases} -\dfrac{2(i+2)}{(2i+3)(2i+5)}, & j = i+2, \\ \dfrac{4}{(2i+1)(2i+3)(2i+5)}, & j = i+1, \\ \dfrac{4(i+1)}{(2i+1)(2i+3)}, & j = i. \end{cases}$$

In the Chebyshev case, A is a upper-triangular matrix with

$$a_{ij} = \begin{cases} (i+1)^2 \pi, & j = i, \\ (i-j)\pi, & j = i+1, i+3, i+5 \cdots \\ (i+j+1)\pi, & j = i+2, i+4, i+6 \cdots, \end{cases}$$

and C is a symmetric penta-diagonal matrix with non-zero elements

$$c_{ii} = \frac{\pi}{2}, \quad i = 0, 1, \cdots, N-1,$$

$$c_{i,i+2} = c_{i+2,i} = -\frac{\pi}{4}, \quad i = 0, 1, \cdots, N-3,$$

$$c_{01} = c_{10} = \frac{\pi}{4}.$$

The matrix equation (6.3.5) can be efficiently solved, in particular, by using the matrix decomposition method. More precisely, we consider the following generalized eigenvalue problem $S\bar{x} = \lambda M\bar{x}$, and let Λ be the diagonal matrix formed by the eigenvalues and E be the matrix formed by the corresponding eigenvectors. Then,

$$SE = ME\Lambda \quad \text{or} \quad E^{\mathrm{T}}S^{\mathrm{T}} = \Lambda E^{\mathrm{T}}M. \tag{6.3.6}$$

Making a change of variable $U = VE^{\mathrm{T}}$ in (6.3.5), we find

$$\frac{1}{4}CVE^{\mathrm{T}}S^{\mathrm{T}} + (A + \beta C)VE^{\mathrm{T}}M = F.$$

We then derive from (6.3.6) that

$$\frac{1}{4}CV\Lambda + (A + \beta C)V = FM^{-1}E^{-T} := G. \tag{6.3.7}$$

Let $\bar{v}_p$ and $\bar{g}_p$ be the p-th column of V and G, respectively. Then (6.3.7) becomes

$$((\frac{1}{4}\lambda_p + \beta)C + A)\bar{v}_p = \bar{g}_p,\ p = 0, 1, \cdots, N-2, \tag{6.3.8}$$

which can be efficiently solved as shown in Sections 3.2 and 3.3.

In summary, after the pre-processing for the computation of the eigenpair (Λ, E) and E^{-1} (in the Legendre case, E is a orthonormal matrix, i.e. $E^{-1} = E^{\mathrm{T}}$), the solution of (6.3.5) consists of three main steps:

1. Compute $G = FM^{-1}E^{-T}$: $N^3 + \mathcal{O}(N^2)$ flops;
2. Solving V from (6.3.8): $\mathcal{O}(N^2)$ flops;
3. Set $U = VE^{\mathrm{T}}$: N^3 flops.

Exercise 6.3

Problem 1 Compute the first eigenvalue of the Bessel's equation

$$\begin{aligned} &-u_{rr} - \frac{1}{r}u_r + \frac{m^2}{r^2}u = \lambda u, \quad r \in (0,1); \\ &u(1) = 0, \quad |u(0)| < \infty. \end{aligned} \tag{6.3.9}$$

For $m = 0$ and $m = 7$, list the results for $N = 8, 16, 32, 64$.

Problem 2 Design an efficient Legendre-Galerkin method for (6.3.1) where $\hat{\Omega} = \{(x, y, z) : a < x^2 + y^2 < b,\ 0 < z < h\}$, assuming F is axisymmetric.

6.4 A fast Poisson Solver using finite differences

Second-order FDM with FFT
Fourth-order compact FDM with FFT
Thomas Algorithm for tridiagonal system
Order of convergence

It is clear that FFT plays an essential role in the efficient implementation of spectral methods, it is also interesting that FFT can be exploited to construct fast algorithms for solving boundary-value problems of elliptic type with finite difference methods (FDM). To illustrate the idea, we again consider the model problem:

$$\begin{cases} u_{xx} + u_{yy} = f(x,y), & \text{in } \Omega, \\ u(x,y) = 0, & \text{on } \partial\Omega, \end{cases} \tag{6.4.1}$$

where $\Omega = \{(x,y) : 0 < x < 1,\, 0 < y < 1\}$. An uniform mesh for the square domain is given by $x_i = ih, y_j = jh, (0 \leqslant i,\, j \leqslant N+1)$, with $h = \frac{1}{N+1}$.

Second-order FDM with FFT

We begin by considering a second-order finite difference approach for the Poisson problem (6.4.1), which is described by Kincaid and Cheney[92]. The solution procedure will be extended to a fourth-order compact scheme in the second part of this section.

A standard five-point scheme, based on the central differencing approach, is given by

$$\frac{v_{i+1,j} - 2v_{ij} + v_{i-1,j}}{h^2} + \frac{v_{i,j+1} - 2v_{ij} + v_{i,j-1}}{h^2} = f_{ij}, \qquad 1 \leqslant i,j \leqslant N, \tag{6.4.2}$$

where $v_{ij} \approx u(x_i, y_j), f_{ij} = f(x_i, y_j)$. The boundary conditions are

$$v_{0,j} = v_{N+1,j} = v_{i,0} = v_{i,N+1} = 0. \tag{6.4.3}$$

The traditional way of proceeding at this juncture is to solve system (6.4.2) by an iterative method. There are N^2 equations and N^2 unknowns. The computational effort to solve this system using, say, successive over-relaxation is $\mathcal{O}(N^3 \log N)$. The alternative approach involving FFT (or Fast Fourier Sine Transform) will bring this effort down to $\mathcal{O}(N^2 \log N)$.

Below we describe how to solve the system (6.4.2) using the Fast Fourier Sine Transform method. A solution of system (6.4.2) will be sought in the following form:

$$v_{ij} = \sum_{k=1}^{N} a_{kj} \sin ik\theta \qquad (0 \leqslant i,\ j \leqslant N+1), \tag{6.4.4}$$

where $\theta = \pi/(N+1)$. Here the numbers a_{kj} are unknowns that we wish to determine. They represent the Fourier sine transform of the function v. Once the a_{kj} have been determined, the fast Fourier sine transform can be used to compute v_{ij} efficiently.

If the v_{ij} from (6.4.4) are substituted into (6.4.2), the result is

$$\begin{aligned} &\sum_{k=1}^{N} a_{kj}[\sin(i+1)k\theta - 2\sin ik\theta + \sin(i-1)k\theta] \\ &+ \sum_{k=1}^{N} \sin ik\theta[a_{k,j+1} - 2a_{kj} + a_{k,j-1}] = h^2 f_{ij}. \end{aligned} \tag{6.4.5}$$

We further introduce the sine transform of f_{ij}:

$$f_{ij} = \sum_{k=1}^{N} \hat{f}_{kj} \sin ik\theta\,. \tag{6.4.6}$$

This, together with a trigonometric identity for (6.4.5), gives

$$\begin{aligned} &\sum_{k=1}^{N} a_{kj}(-4\sin ik\theta)\sin^2(k\theta/2) + \sum_{k=1}^{N} \sin ik\theta(a_{k,j+1} - 2a_{kj} + a_{k,j-1}) \\ &\qquad = h^2 \sum_{k=1}^{n} \hat{f}_{kj} \sin ik\theta\,. \end{aligned} \tag{6.4.7}$$

Therefore, we can deduce from (6.4.7) that

$$a_{kj}\left(-4\sin^2\frac{k\theta}{2}\right) + a_{k,j+1} - 2a_{kj} + a_{k,j-1} = h^2 \hat{f}_{kj} \tag{6.4.8}$$

The above equation appears at first glance to be another system of N^2 equations in N^2 unknowns, which is only slightly different from the original system (6.4.2). But closer inspection reveals that in (6.4.8), k can be held fixed, and the resulting system of N equations can be easily and *directly* solved since it is tridiagonal. Thus for fixed k, the unknowns in (6.4.8) form a vector $[a_{k1}, \cdots, a_{kN}]^{\mathrm{T}}$ in $\mathbb{R}^N$. The procedure used above has *decoupled* the original system of N^2 equations into N systems of N equations each. A tridiagonal system of N equations can be solved in $\mathcal{O}(N)$ operations

(in fact, fewer than $10N$ operations are needed). Thus, we can solve N tridiagonal systems at a cost of $10N^2$. The fast Fourier sine transform uses $\mathcal{O}(N \log N)$ operations on a vector with N components. Thus, the total computational burden in the fast Poisson method is $\mathcal{O}(N^2 \log N)$.

Fourth-order compact FDM with FFT

We now extend the fast solution procedure described above to deal with a more accurate finite difference approach, namely, 4th-order compact finite difference method for the Poisson problem (6.4.1). The finite difference method (6.4.2) has an overall $\mathcal{O}(h^2)$ approximation accuracy. Using a compact 9-point scheme, the accuracy can be improved to 4th-order, and the resulting system can be also solved with $\mathcal{O}(N^2 \log N)$ operations.

In the area of finite difference methods, it has been discovered that the second-order central difference approximations (such as (6.4.2)), when being used for solving the convection-diffusion equations often suffer from computational instability and the resulting solutions exhibit nonphysical oscillations; see e.g. [133]. The upwind difference approximations are computationally stable, although only first-order accurate, and the resulting solutions exhibit the effects of artificial viscosity. The second-order upwind methods are no better than the first-order upwind difference ones for convection-dominated problems. Moreover, the higher-order finite difference methods of conventional type do not allow direct iterative techniques. An exception has been found in the high order finite difference schemes of compact type that are computationally efficient and stable and yield highly accurate numerical solutions [39, 78, 151, 173].

Assuming a uniform grid in both x and y directions, we number the grid points (x_i, y_j), (x_{i+1}, y_j), (x_i, y_{j+1}), (x_{i-1}, y_j), (x_i, y_{j-1}), (x_{i+1}, y_{j+1}), (x_{i-1}, y_{j+1}), (x_{i-1}, y_{j-1}), (x_{i+1}, y_{j-1}) as 0, 1, 2, 3, 4, 5, 6, 7, 8, respectively (see Fig. 6.1). In writing the FD approximations a single subscript k denotes the corresponding function value at the grid point numbered k.

We first derive the 4th-order compact scheme. A standard Taylor expansion for the central differencing gives

$$\begin{aligned} \frac{u_1 + u_3 - 2u_0}{h^2} &= (u_{xx})_0 + \frac{h^2}{12}(u_{xxxx})_0 + \mathcal{O}(h^4), \\ \frac{u_2 + u_4 - 2u_0}{h^2} &= (u_{yy})_0 + \frac{h^2}{12}(u_{yyyy})_0 + \mathcal{O}(h^4). \end{aligned} \tag{6.4.9}$$

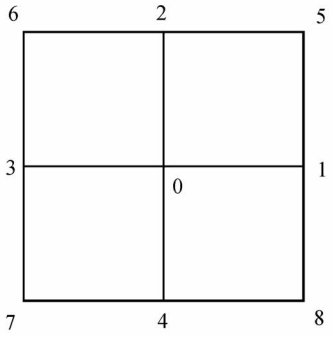

Figure 6.1 The mesh stencil for the compact scheme

These results, together with the Poisson equation in (6.4.1), gives

$$\begin{aligned}&\frac{u_1+u_3-2u_0}{h^2}+\frac{u_2+u_4-2u_0}{h^2}\\&=f_0+\frac{h^2}{12}(u_{xxxx}+u_{yyyy})_0+\mathcal{O}(h^4).\end{aligned}\tag{6.4.10}$$

In order to obtain a 4th-order accuracy, a second-order approximation for the term $u_{xxxx}+u_{yyyy}$ is needed. However, direct central differencing approximations for u_{xxxx} and u_{yyyy} with $\mathcal{O}(h^2)$ accuracy requires stencils outside the 9-points in the box Fig. 6.1. To fix it, we again use the governing equation for u:

$$\begin{aligned}&(u_{xxxx}+u_{yyyy})_0=(\nabla^4u)_0-2(u_{xxyy})_0\\=&(\nabla^2f)_0-2(u_{xxyy})_0\\=&(f_1+f_3-2f_0)/h^2+(f_2+f_4-2f_0)/h^2-2(u_{xxyy})_0+\mathcal{O}(h^2).\end{aligned}\tag{6.4.11}$$

It can be shown that the mixed derivative u_{xxyy} can be approximated by the 9-points stencil with $\mathcal{O}(h^2)$ truncation errors:

$$\begin{aligned}&\frac{1}{h^2}\left[\frac{u_5+u_6-2u_2}{h^2}-2\frac{u_1+u_3-2u_0}{h^2}+\frac{u_7+u_8-2u_4}{h^2}\right]\\=&((u_{xx})_2-2(u_{xx})_0+(u_{xx})_4)/h^2+\mathcal{O}(h^2)\\=&(u_{xxyy})_0+\mathcal{O}(h^2).\end{aligned}\tag{6.4.12}$$

Using the above results, we obtain a 4th-order finite difference scheme using the compact stencils:

$$\frac{u_1+u_3-2u_0}{h^2}+\frac{u_2+u_4-2u_0}{h^2}$$

$$
\begin{aligned}
=& f_0 + \frac{1}{12}\left[f_1 + f_3 - 2f_0 + f_2 + f_4 - 2f_0\right] \\
&- \frac{1}{6}\left[\frac{u_5 + u_6 - 2u_2}{h^2} - 2\frac{u_1 + u_3 - 2u_0}{h^2} + \frac{u_7 + u_8 - 2u_4}{h^2}\right].
\end{aligned} \tag{6.4.13}
$$

Using the sine expansion of the form (6.4.4): $u_{ij} = \sum_{k=1}^{N} a_{kj}\sin(ik\theta), \theta = \pi/(N+1)$, we can obtain

$$
\begin{aligned}
u_5 + u_6 - 2u_2 &= \sum_{k=1}^{N} a_{k,j+1}\left(-4\sin(ik\theta)\right)\sin^2(k\theta/2), \\
u_1 + u_3 - 2u_0 &= \sum_{k=1}^{N} a_{k,j}\left(-4\sin(ik\theta)\right)\sin^2(k\theta/2), \\
u_7 + u_8 - 2u_4 &= \sum_{k=1}^{N} a_{k,j-1}\left(-4\sin(ik\theta)\right)\sin^2(k\theta/2).
\end{aligned}
$$

These results, together with the results similar to f_j's, yield an equivalent form for the finite difference scheme (6.4.13):

$$
\begin{aligned}
&\sum_{k=1}^{N} a_{kj}\big(-4\sin(ik\theta)\big)\sin^2\frac{k\theta}{2} + \sum_{k=1}^{N}\big(a_{k,j+1} - 2a_{k,j} + a_{k,j-1}\big)\sin(ik\theta) \\
=& h^2\sum_{k=1}^{N}\hat{f}_{kj}\sin(ik\theta) + \frac{h^2}{12}\sum_{k=1}^{N}\hat{f}_{kj}\big(-4\sin(ik\theta)\big)\sin^2\frac{k\theta}{2} \\
&+\frac{h^2}{12}\sum_{k=1}^{N}\big(\hat{f}_{k,j+1} - 2\hat{f}_{k,j} + \hat{f}_{k,j-1}\big)\sin(ik\theta) \\
&- \frac{1}{6}\sum_{k=1}^{N}\left(a_{k,j+1} - 2a_{k,j} + a_{k,j-1}\right)\big(-4\sin(ik\theta)\big)\sin^2\frac{k\theta}{2},
\end{aligned} \tag{6.4.14}
$$

where $\hat{f}_{kj}$ is defined by (6.4.6). Grouping the coefficients of $\sin(ik\theta)$ gives

$$
\begin{aligned}
&-4\sin^2(k\theta/2)a_{k,j} + \big(a_{k,j+1} - 2a_{k,j} + a_{k,j-1}\big) \\
=& h^2\hat{f}_{k,j} + \frac{h^2}{12}\left\{-4\sin^2(k\theta/2)\hat{f}_{k,j} + \big(\hat{f}_{k,j+1} - 2\hat{f}_{k,j} + \hat{f}_{k,j-1}\big)\right\} \\
&-\frac{1}{6}\big(a_{k,j+1} - 2a_{k,j} + a_{k,j+1}\big)\left(-4\sin^2(k\theta/2)\right).
\end{aligned}
$$

Finally, we obtain a tridiagonal system for the coefficients a_{kj} with each fixed k:

$$\left(1-\frac{2}{3}\sin^2\frac{k\theta}{2}\right)a_{k,j+1}+\left(-2-\frac{8}{3}\sin^2\frac{k\theta}{2}\right)a_{k,j}+\left(1-\frac{2}{3}\sin^2\frac{k\theta}{2}\right)a_{k,j-1}$$
$$=\frac{h^2}{12}\left[\hat{f}_{k,j+1}+\left(10-4\sin^2(k\theta/2)\right)\hat{f}_{k,j}+\hat{f}_{k,j-1}\right]. \tag{6.4.15}$$

Thomas Algorithm for tridiagonal system

Consider the tridiagonal system

$$\begin{bmatrix} b_1 & c_1 & & & & & & \\ a_2 & b_2 & c_2 & & & & & \\ & \cdot & \cdot & \cdot & & & & \\ & & a_i & b_i & c_i & & & \\ & & & \cdot & \cdot & \cdot & & \\ & & & & a_{N-1} & b_{N-1} & c_{N-1} & \\ & & & & & a_N & b_N & \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_i \\ \vdots \\ v_{N-1} \\ v_N \end{bmatrix}=\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_i \\ \vdots \\ d_{N-1} \\ d_N \end{bmatrix}, \tag{6.4.16}$$

where a_i, b_i, c_i and d_i are given constants. All terms in the above matrix, other than those shown, are zero. The Thomas algorithm for solving (6.4.16) consists of two parts. First, the tridiagonal system (6.4.16) is manipulated into the form

$$\begin{bmatrix} 1 & c_1' & & & & & & \\ & 1 & c_2' & & & & & \\ & & \cdot & \cdot & \cdot & & & \\ & & & 1 & c_i' & & & \\ & & & & \cdot & \cdot & \cdot & \\ & & & & & 1 & c_{N-1}' & \\ & & & & & & 1 & \end{bmatrix}\begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_N \end{bmatrix}=\begin{bmatrix} d_1' \\ \vdots \\ d_i' \\ \vdots \\ d_N' \end{bmatrix},$$

i.e. the coefficients a_i have been eliminated and the coefficients b_i normalized to unity. For the first equation,

$$c_1'=\frac{c_1}{b_1}, \qquad d_1'=\frac{d_1}{b_1}, \tag{6.4.17}$$

and for the general equation

$$c'_i = \frac{c_i}{b_i - a_i c'_{i-1}},\ 1 < i < N, \qquad d'_i = \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}},\quad 1 < i \leqslant N\ . \tag{6.4.18}$$

The second stage consists of a backward-substitution:

$$v_N = d'_N \qquad \text{and} \qquad v_i = d'_i - v_{i+1} c'_i\ ,\quad i = N-1, \cdots, 1. \tag{6.4.19}$$

The Thomas algorithm is particularly economical; it requires only $5N-4$ operations (multiplications and divisions). But to prevent ill-conditioning (and hence round-off contamination) it is necessary that

$$|b_i| > |a_i| + |c_i|\ .$$

The procedures of solving the finite difference schemes with FFT described in this section typically generate tridiagonal systems of equations that can be solved efficiently using the Thomas algorithm.

Order of convergence

If a finite difference scheme is given, one way to find its order of convergence is to use the Taylor expansion to find its truncation error. Another way is to find the order by doing some simple numerical tests. The procedure is described below:

(a) Pick up a single test equation in a simple geometry (say, square domain), such that the exact solution is known;

(b) Solve the test equation using the given finite difference scheme with at least three sets of mesh sizes: h_1, h_2, and h_3;

(c) Since the exact solution is known, the errors (L^1, L^∞ or L^c) associated with h_1, h_2 and h_3 can be obtained easily. They are denoted by e_1, e_2 and e_3, respectively.

Having the above steps, we are able to find the order of convergence as follows. Assume the leading term of the error is

$$e \approx C h^\alpha,$$

for some constant C and α. Here α is the desired order of convergence. Since

$$e_i \approx C h_i^\alpha,\quad i = 1, 2, 3,$$

we have

$$\frac{e_1}{e_2} \approx \left(\frac{h_1}{h_2}\right)^{\alpha}, \quad \left(\frac{e_1}{e_2}\right) \approx \left(\frac{h_2}{h_3}\right)^{\alpha}.$$

This gives

$$\alpha_1 = \log\left(\frac{e_1}{e_2}\right) \Big/ \log\left(\frac{h_1}{h_2}\right), \quad \alpha_2 = \log\left(\frac{e_2}{e_3}\right) \Big/ \log\left(\frac{h_2}{h_3}\right).$$

If the two α's obtained above are very close to each other, then the value of α gives a good approximation of the convergence order. If they are not close, a smaller mesh h_4 should be used, and the value

$$\alpha_3 = \log\left(\frac{e_3}{e_4}\right) \Big/ \log\left(\frac{h_3}{h_4}\right)$$

should be compared with α_2 and α_1.

In practice, we choose $h_{i+1} = h_i/2$, for $i = 1, 2, \cdots$. Namely, we always halve the mesh size, and to observe the change of the resulting errors. In this case, the formula for computing the convergence order becomes

$$\text{order} = \log_2\left(\frac{\texttt{error}(h)}{\texttt{error}(h/2)}\right). \tag{6.4.20}$$

We close this section by pointing out that we presented only a simple case for the many versions of fast Poisson solvers. There are many relevant papers on either algorithm development or theoretical justifications; see e.g. [14], [128], [140], [143], [153].

Exercise 6.4.

Problem 1 Consider the Poisson problem

$$\begin{cases} u_{xx} + u_{yy} = 2e^{x+y}, & (x, y) \in \Omega = (0, 1) \times (0, 1), \\ u|_{\partial\Omega} = e^{x+y}. \end{cases} \tag{6.4.21}$$

The exact solution for the above problem is $u(x, y) = e^{x+y}$. Solve the above problem by using the 5-point finite difference scheme (6.4.2) and list the L^1-errors and the L^∞ errors. Demonstrate that the scheme (6.4.2) is of second-order accuracy.

Problem 2 Solve the Poisson problem

$$\begin{aligned} &u_{xx} + u_{yy} = -2\pi^2 \sin(\pi x)\sin(\pi y), \qquad 0 < x, y < 1, \\ &u(x, y) = 0, \qquad \text{on the boundary}, \end{aligned} \tag{6.4.22}$$

using the 2nd-order finite difference method (6.4.2) and the fast sine transform, with $N = 10, 20$ and 40. The exact solution is $u(x, y) = \sin(\pi x)\sin(\pi y)$. Plot the error function.

Problem 3 Repeat the above problem using the 4th-order compact scheme (6.4.13) and the fast sine transform. Moreover, by comparing the L^1 errors for the three N's to show that the numerical method used is of 4-th order accuracy.

Chapter 7

Some applications in multi-dimensions

Contents

7.1 Spectral methods for wave equations 257
7.2 Laguerre-Hermite method for Schrödinger equations 264
7.3 Spectral approximation of the Stokes equations 276
7.4 Spectral-projection method for Narier-stokes equations . . . 282
7.5 Axisymmetric flows in a cylinder 288

We present in this chapter applications of the spectral methods to several problems in multi-dimensional domains. In Section 7.1, we present two examples of two-dimensional time-dependent scalar advection equation in Cartesian coordinates. In Section 7.2, we present a fourth-order time-splitting spectral method for the numerical simulation of the Gross-Pitaevskii equation (GPE) which describes a Bose-Einstein condensate (BEC) at temperatures T much smaller than the critical temperature T_c. The scheme preserves all essential features of the GPE. The remaining three sections are concerned with topics in computational fluid dynamics. In Section 7.3, we present a spectral approximation for the Stokes equations. In Section 7.4, we will describe two robust and accurate projection type schemes and the related full discretization schemes with a spectral-Galerkin discretization in space. Finally, In Section 7.5, we apply the spectral-projection scheme to simulate an incompressible flow inside a cylinder.

7.1 Spectral methods for wave equations

Linear advection problems
Numerical algorithms
Grid mapping for the Chebyshev collocation method
Numerical results

In this section, we will present two examples of two-dimensional time-dependent scalar advection equation in Cartesian coordinates. It is instructive to see how one should implement the spectral method for this class of problems. The first example is the pure linear advection equation and the second is the rotational wave equation. These two equations differ only by the fact that the wave velocity is constant for the first one while it depends on the spatial variables in the second case. They will be used to demonstrate some practical issues in implementing the spectral methods such as smoothing, filtering and Runge-Kutta time stepping discussed in previous chapters. We will focus on the Fourier and Chebyshev collocation methods. Some research papers relevant to this section include [22], [29], [118].

Linear advection problems

Example 7.1.1 *The first example is*

$$\frac{\partial U}{\partial t} + a_x \frac{\partial U}{\partial x} + a_y \frac{\partial U}{\partial y} = 0, \quad (x, y) \in (-1, 1)^2, \quad t > 0, \tag{7.1.1}$$

where $U(x, y, t)$ is a function of two spatial variables (x, y) and time t, and the wave speeds $a_x \geqslant 0$ and $a_y \geqslant 0$ are constant. We assume a periodic boundary condition in the y direction and impose a Dirichlet inflow boundary condition at $x = -1$.

$$U(-1, y, t) = 0 \qquad -1 \leqslant y \leqslant 1. \tag{7.1.2}$$

Equation (7.1.1) models the propagation of a signal initially at rest through its evolution in space and time. The solution of the partial differential equation (7.1.1) is

$$U(\vec{x}, t) = U_0(\vec{x} - \vec{a}t), \quad \vec{x} = (x, y), \quad \vec{a} = (a_x, a_y), \tag{7.1.3}$$

where $U_0(\vec{x})$ is an initial function (signal) at $t = 0$, that is, the shape of the signal moving in the (x, y)-plane with a constant speed according to the given wave velocity vector (a_x, a_y) and elapsed time t (see Figure 7.2). Since the physical domain in x is finite, the signal will move out from the right boundary in some finite time.

Example 7.1.2 *The second example is a linear rotational problem*

$$\frac{\partial U}{\partial t} + y\frac{\partial U}{\partial x} - x\frac{\partial U}{\partial y} = 0, \quad (x, y) \in (-1, 1)^2, \quad t > 0, \tag{7.1.4}$$

where $U(x, y, t)$ is a function of two spatial variables (x, y) and time t. The boundary conditions are periodical in the y direction, but no boundary condition is imposed at $x = \pm 1$.

The wave velocity $(a_x, a_y) = (y, -x)$ in (7.1.4) is the tangential velocity at the circumference of a circle. In this case, a circular axis-symmetric Gaussian pulse centered at $(C_x, C_y) = (0, 0)$ (see the initial conditions below) will simply rotate about its central axis without any displacement in either direction. Hence the theoretical solution of (7.1.4) will remain unchanged and appeared stationary at all time for the circular Gaussian pulse centered at $(C_x, C_y) = (0, 0)$. It is an excellent test which displays the dispersive and dissipative nature of a long time integration by a given numerical algorithm.

`Initial condition` For both examples above, the initial condition is specified as a smooth circular Gaussian function in the form of

$$U(x, y, t = 0) = U_0(x, y) = \begin{cases} 0, & L > 1, \\ e^{-\alpha |L|^\gamma}, & L \leqslant 1, \end{cases} \tag{7.1.5}$$

where $L = \sqrt{(x - C_x)^2 + (y - C_y)^2}/R$ with (C_x, C_y) the center of the Gaussian function and R a given parameter that control the compact support of the Gaussian function; $\alpha = -\log(\epsilon)$ with ϵ the machine roundoff errors; and γ is the order of the Gaussian function.

In an actual implementation, R is chosen as

$$R = \beta \min(L_x, L_y), \quad \beta \leqslant 1, \tag{7.1.6}$$

where L_x and L_y are the length of the physical domain in the x and y direction, respectively. In our computations, we choose $L_x = L_y = 2$. For the linear advection problem, we will take $(a_x, a_y) = (1, 2.5)$, $(C_x, C_y) = (-1 + L_x/2, -1 + L_y/2)$, $\beta = 0.25$ and $\gamma = 8$. The final time is set at $t_f = 1.5$. For Example 7.1.2, we will center the Gaussian pulse in the middle of the physical domain with $(C_x, C_y) = (0, 0)$ while keeping all the other parameters unchanged.

Numerical algorithms

Since both problems are periodic in y, it is natural to employ the Fourier col-

location method in the y direction. In the x direction, both the Legendre and the Chebyshev collocation methods can be used. We will use the Chebyshev collocation method in this section, partly due to the fact that with the Chebyshev method the Fast Cosine Transform method can be employed.

To solve the PDE numerically, we need to replace the continuous differentiation operators by appropriate discretized counterparts. If we denote the Fourier differentiation matrix by D_y^f and the Chebyshev differentiation matrix by D_x^c, where the subscript denotes the coordinate direction on which the differentiation operates, the discretized version of (7.1.1) can be written in the matrix form

$$\frac{\partial \vec{U}}{\partial t} + a_x D_x^c \vec{U} + a_y D_y^f \vec{U} = 0, \tag{7.1.7}$$

where the two-dimensional array $\vec{U} = U(x_i, y_j, t)$, with $x_i = \cos(\pi i/N_x), i = 0 \cdots N_x$ being the Chebyshev-Gauss-Lobatto collocation points and $y_j = -1 + 2j/N_y, j = 0 \cdots N_y - 1$ being the Fourier collocation points. Since the domain limit in y is $[-1, 1)$ instead of the classical $[0, 2\pi)$, the Fourier operator D_y^f in (7.1.7) is scaled by a factor of π. The equation (7.1.7) is a system of ordinary differential equations which can be advanced by any standard stable high-order Runge-Kutta method. We used the third-order TVD Runge-Kutta scheme to solve the ODE system[149]:

$$\begin{aligned}
\vec{U}^1 &= \vec{U}^n + \Delta t \mathcal{L}(\vec{U}^n), \\
\vec{U}^2 &= \frac{1}{4}(3\vec{U}^n + \vec{U}^1 + \Delta t \mathcal{L}(\vec{U}^1)), \\
\vec{U}^{n+1} &= \frac{1}{3}(\vec{U}^n + 2\vec{U}^2 + 2\Delta t \mathcal{L}(\vec{U}^2)),
\end{aligned} \tag{7.1.8}$$

where $\mathcal{L} = -(a_x D_x^c + a_y D_y^f)$ is the spatial operator, and $\vec{U}^1$and $\vec{U}^2$ are two temporary arrays at the intermediate Runge-Kutta stages. Notice that this Runge-Kutta scheme requires only one temporary array to be allocated since$\vec{U}^1$ can be overwritten by $\vec{U}^2$ in the second stage. The scheme has been shown to be stable for

$$\text{CFL} = \Delta t \max_{i,j}(|a_x|/\Delta x_i + |a_y|/\Delta y_j) \leqslant 1. \tag{7.1.9}$$

In our computations, we used CFL=1. Furthermore, at each Runge-Kutta stage, an appropriate boundary condition, say $\vec{U}(-1, y_j, t) = 0$, should be imposed if it is prescribed.

As discussed in Section 5.5, filters may be needed to stabilize spectral computations. For Example 7.1.2, a 16th-order exponential filter is used in the computations.

Grid mapping for the Chebyshev collocation method

A major disadvantage in using the Chebyshev collocation methods is that when a k-th derivative (in space) is treated explicitly in a time discretization scheme, it leads to a very restrictive CFL condition $\Delta t \approx \mathcal{O}(N^{-2k})$. As discussed at the end of Section 2.3, this is due to the clustering of the Chebyshev points at the boundaries. In order to alleviate the time step restriction, Kosloff and Tal-Ezer[93] devised a grid mapping technique that maps the original Chebyshev-Gauss-Lobatto points into another set of collocation points. The mapping has the form of

$$x = g(\xi, \alpha) = \frac{\sin^{-1}(\alpha\xi)}{\sin^{-1}\alpha}, \tag{7.1.10}$$

where ξ and x are the original and mapped Chebyshev collocation points, respectively. The main effect of this mapping is that the minimum spacing is increased from $\Delta\xi \approx \mathcal{O}(N^{-2})$ in the original Chebyshev grid to $\Delta x \approx \mathcal{O}(N^{-1})$ in the new mapped Chebyshev grid as the mapping parameter $\alpha \to 1$.

Under the mapping (7.1.10), the differentiation matrix D_b becomes

$$\mathcal{D}_b = \mathcal{M} D_b, \tag{7.1.11}$$

where $\mathcal{M}$ is a diagonal matrix with elements

$$\mathcal{M}_{ii} = g'(\xi_i, \alpha)^{-1} = g'(y) = \frac{\alpha}{\arcsin(\alpha)} \frac{1}{\sqrt{1-(\alpha y)^2}}. \tag{7.1.12}$$

However, in order to retain the spectral accuracy of the mapped Chebyshev collocation method, the parameter α cannot be chosen arbitrarily. It had been shown that if α is chosen as

$$\alpha = \alpha(N, \epsilon) = \text{sech}\left(|\ln \epsilon|/N\right), \tag{7.1.13}$$

then the approximation error is roughly ϵ. Note that α is not a constant but a function of N. By choosing ϵ to be of machine epsilon, the error of the grid mapping is essentially guaranteed to be harmless.

A natural question is what will be the extra work of grid mapping for the mapped Chebyshev collocation method? In Figure 7.1, the eigenvalue spectrum of the original and the mapped Chebyshev differentiation matrix with the Dirichlet boundary condition are shown with $N = 64$ collocation points. It is observed that the largest eigenvalue of the mapped Chebyshev differentiation matrix $\mathcal{D}_b$ is substantially smaller

than the one computed with the original unmapped counterpart D_b. Intuitively, we can expect that for the k-th derivatives the mapped Chebyshev method will

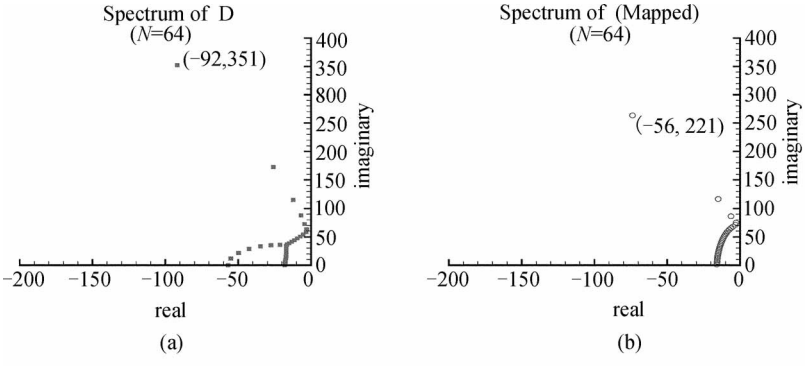

Figure 7.1

(a) The eigenvalue spectrum of the original;

(b) the mapped Chebyshev differentiation matrix with the Dirichlet boundary condition.

- reduce the roundoff error from $\mathcal{O}(N^{2k}\epsilon)$ to $\mathcal{O}(N^k\epsilon)$ as shown in Table 7.1;
- reduce the spectral radius of the differentiation matrix from $\mathcal{O}(N^{2k})$ to $\mathcal{O}(N^k)$ asymptotically for D_b and $\mathcal{D}_b$ respectively, as shown in Table 7.2.

Table 7.1 Absolute maximum error for the second derivative of $\sin(2x)$

N	No Mapping	with Mapping
32	0.47E-09	0.20E-09
64	0.62E-08	0.20E-08
128	0.71E-07	0.13E-07
256	0.35E-05	0.21E-06
512	0.98E-05	0.33E-06
1024	0.13E-02	0.21E-05

Table 7.2 The spectral radius λ of D_b and $\mathcal{D}_b$ with $k = 1$

N	$\lambda(D_b)$	Growth Rate	$\lambda(\mathcal{D}_b)$	Growth Rate
32	91.6		80.8	
64	263.8	2	230.4	1.50
128	1452.7	2	555.4	1.27
256	5808.4	2	1219.1	1.13
512	23231.3	2	2553.5	1.07
1024	92922.8	2	5225.8	1.03

In summary, the spectral algorithm consists of the following features:

- Spatial Algorithm:
  Chebyshev and Fourier collocation methods.
  - Differentiation and smoothing operations are done via an optimized library PseudoPack (Costa & Don);
  - 16-th order exponential filters are used for the differentiation and solution smoothing when needed;
  - The Kosloff-Tal-Ezer mapping is used for accuracy and stability enhancement for the Chebyshev collocation methods.
- Temporal Algorithm:
  Third-order explicit TVD Runge-Kutta method.

Numerical results

The numerical results for Examples 7.1.1 and 7.1.2 are shown in Figures 7.2 and 7.3, respectively, computed by using a 256×256 resolution. As depicted in Figure 7.2, the Gaussian pulse, initially at the upper left corner, moves diagonally down and partially exits the physical domain at the later time. For Example 7.1.2, an one-dimensional cut through the center of the solution at time $t = 0$ (square symbol) and $t = 1.5$ (circle symbol) is shown in Figure 7.4. The symbols are overlapped with each other since the difference between them is on the order of 10^{-7} or smaller.

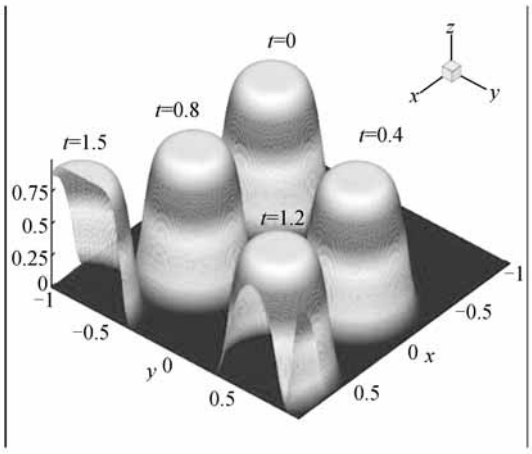

Figure 7.2 Solution of Example 7.1.1 with the Gaussian pulse at various time.

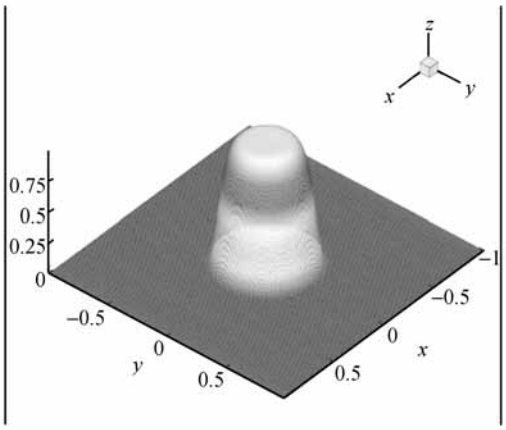

Figure 7.3 Solution of Example 7.1.2 with the Gaussian pulse at $t = 1.5$.

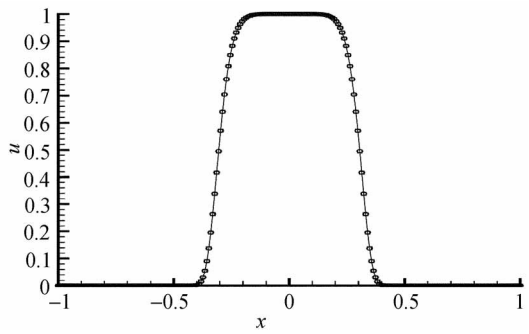

Figure 7.4 an one-dimensional cut through the center at time $t = 0$ (square) and $t = 1.5$ (circle).

For both examples, the shape of the pulse remain sharp and well defined without any distortion and has minimal classical dispersive and dissipative behavior we may otherwise observe in a finite difference or a finite element computation.

The FORTRAN code can be found in

`http://www.math.hkbu.edu.hk/~ttang/PGteaching;`

it is written in FORTRAN 90/95, and is based on the PseudoPack library co-developed by Wai Sun Don and Bruno Costa. Many important and optimized subroutines for computing differentiation by the Fourier, Chebyshev and Legendre collocation methods using various advance algorithms are incorporated into the library. Some further information of the PseudoPack can be found in Appendix C of this book. Readers

who are interested in the library can visit

`http://www.cfm.brown.edu/people/wsdon/home.html`

Exercise 7.1.
Problem 1 Consider the mapping (7.1.10). Show that with this mapping the minimum spacing can be increased from $\Delta\xi \approx \mathcal{O}(N^{-2})$ in the original Chebyshev grid to $\Delta x \approx \mathcal{O}(N^{-1})$ in the new mapped Chebyshev grid as the mapping parameter $\alpha \to 1$.

7.2 Laguerre-Hermite method for Schrödinger equations

The Gross-Pitaevskii equation (GPE)
Hermite pseudospectral method for the 1-D GPE
Two-dimensional GPE with radial symmetry
Three-dimensional GPE with cylindrical symmetry
Numerical results

The nonlinear Schrödinger equation plays an important role in many fields of mathematical physics. In particular, at temperatures T much smaller than the critical temperature T_c, a Bose-Einstein condensate (BEC) is well described by the macroscopic wave function $\psi = \psi(\mathbf{x}, t)$ whose evolution is governed by a self-consistent, mean field nonlinear Schrödinger equation (NLSE) known as the Gross-Pitaevskii equation (GPE)[67, 129]. We present in this section a fourth-order time-splitting spectral method for the numerical simulation of BEC. The scheme preserves all essential features of the GPE, such as conservative, time reversible and time transverse invariants, while being explicit , unconditionally stable, and spectrally accurate in space and fourth-order accurate in time.

The Gross-Pitaevskii equation (GPE)

We consider the non-dimensional Gross-Pitaevskii equation in the form

$$i\,\frac{\partial\psi(\mathbf{x},t)}{\partial t} = -\frac{1}{2}\nabla^2\psi(\mathbf{x},t) + V(\mathbf{x})\psi(\mathbf{x},t) + \beta\,|\psi(\mathbf{x},t)|^2\psi(\mathbf{x},t), \qquad (7.2.1)$$

where the unknown is the complex wave function ψ, $i = \sqrt{-1}$, β is a positive constant and

$$V(\mathbf{x}) = \left(\gamma_x^2 x^2 + \gamma_y^2 y^2 + \gamma_z^2 z^2\right)/2 \qquad (7.2.2)$$

is the trapping potential. There are two typical extreme regimes between the trap frequencies: (i) $\gamma_x = 1$, $\gamma_y \approx 1$ and $\gamma_z \gg 1$, it is a disk-shaped condensation; (ii) $\gamma_x \gg 1$, $\gamma_y \gg 1$ and $\gamma_z = 1$, it is a cigar-shaped condensation. Following the procedure used in [9], [98], the disk-shaped condensation can be effectively modeled

by a 2-D GPE. Similarly, a cigar-shaped condensation can be reduced to a 1-D GPE. Hence, we shall consider the GPE in d-dimension ($d = 1, 2, 3$):

$$\begin{aligned} & i\,\frac{\partial \psi(\mathbf{x},t)}{\partial t} = -\frac{1}{2}\nabla^2\psi + V_d(\mathbf{x})\psi + \beta_d\,|\psi|^2\psi, \quad \mathbf{x}\in\mathbb{R}^d, \\ & \psi(\mathbf{x},0) = \psi_0(\mathbf{x}), \qquad \mathbf{x}\in\mathbb{R}^d, \end{aligned} \tag{7.2.3}$$

with

$$\beta_d = \begin{cases} \sqrt{\gamma_x\gamma_y}/2\pi, \\ \sqrt{\gamma_z/2\pi}, \\ 1, \end{cases} \qquad V_d(\mathbf{x}) = \begin{cases} \gamma_z^2 z^2/2, & d=1, \\ \left(\gamma_x^2x^2+\gamma_y^2y^2\right)/2, & d=2, \\ \left(\gamma_x^2x^2+\gamma_y^2y^2+\gamma_z^2z^2\right)/2, & d=3, \end{cases}$$

where $\gamma_x > 0$, $\gamma_y > 0$ and $\gamma_z > 0$ are constants. It is easy to check that this equation is conservative in the sense that

$$\|\psi(\cdot,t)\|^2 := \int_{\mathbb{R}^d} |\psi(\mathbf{x},t)|^2\,\mathrm{d}\mathbf{x} \equiv \int_{\mathbb{R}^d} |\psi_0(\mathbf{x})|^2\,\mathrm{d}\mathbf{x}. \tag{7.2.4}$$

We normalize the initial condition to be

$$\int_{\mathbb{R}^d} |\psi_0(\mathbf{x})|^2\,\mathrm{d}\mathbf{x} = 1. \tag{7.2.5}$$

Moreover, the GPE is time reversible and time transverse invariant (cf. [9]). Hence, it is desirable that the numerical scheme satisfies these properties as well.

For the time discretization, we shall use the fourth-order splitting scheme (1.6.29). To this end, we rewrite the GPE (7.2.3) in the form

$$u_t = f(u) = -iAu - iBu, \qquad u(t_0) = u_0, \tag{7.2.6}$$

$$A\psi = \beta_d\,|\psi(\mathbf{x},t)|^2\psi(\mathbf{x},t), \quad B\psi = -\frac{1}{2}\nabla^2\psi(\mathbf{x},t) + V_d(\mathbf{x})\psi(\mathbf{x},t). \tag{7.2.7}$$

Thus, the key for an efficient implementation of (1.6.29) is to solve efficiently the following two sub-problems:

$$i\,\frac{\partial \psi(\mathbf{x},t)}{\partial t} = A\psi(\mathbf{x},t), \qquad \mathbf{x}\in\mathbb{R}^d, \tag{7.2.8}$$

and

$$i\,\frac{\partial \psi(\mathbf{x},t)}{\partial t} = B\psi(\mathbf{x},t), \quad \mathbf{x}\in\mathbb{R}^d; \qquad \lim_{|\mathbf{x}|\to+\infty}\psi(\mathbf{x},t) = 0, \tag{7.2.9}$$

where the operators A and B are defined by (7.2.7). Multiplying (7.2.8) by $\overline{\psi(\mathbf{x},t)}$, we find that the ordinary differential equation (7.2.8) leaves $|\psi(\mathbf{x},t)|$ invariant in t. Hence, for $t \geqslant t_s$ (t_s is any given time), (7.2.8) becomes

$$i\,\frac{\partial\psi(\mathbf{x},t)}{\partial t} = \beta_d\,|\psi(\mathbf{x},t_s)|^2\psi(\mathbf{x},t),\ t \geqslant t_s, \qquad \mathbf{x}\in\mathbb{R}^d, \tag{7.2.10}$$

which can be integrated **exactly**, i.e.,

$$\psi(\mathbf{x},t) = e^{-i\beta_d|\psi(\mathbf{x},t_s)|^2(t-t_s)}\psi(\mathbf{x},t_s), \qquad t\geqslant t_s, \quad \mathbf{x}\in\mathbb{R}^d. \tag{7.2.11}$$

Thus, it remains to find an efficient and accurate scheme for (7.2.9). We shall construct below suitable spectral basis functions which are eigenfunctions of B so that $e^{-iB\Delta t}\psi$ can be evaluated exactly (which is necessary for the final scheme to be time reversible and time transverse invariant). Hence, the only time discretization error of the corresponding time splitting method (1.6.29) is the splitting error, which is fourth order in Δt. Furthermore, the scheme is explicit, time reversible and time transverse invariant, and as we shall show below, it is also unconditionally stable.

Hermite pseudospectral method for the 1-D GPE

In the 1-D case, Eq. (7.2.9) collapses to

$$i\,\frac{\partial\psi}{\partial t} = -\frac{1}{2}\frac{\partial^2\psi}{\partial z^2} + \frac{\gamma_z^2 z^2}{2}\psi, \quad z\in\mathbb{R}, t>0; \qquad \lim_{|z|\to+\infty}\psi(z,t) = 0,\ t\geqslant 0, \tag{7.2.12}$$

with the normalization (7.2.5)

$$\|\psi(\cdot,t)\|^2 = \int_{-\infty}^{\infty}|\psi(z,t)|^2\mathrm{d}z \equiv \int_{-\infty}^{\infty}|\psi_0(z)|^2\mathrm{d}z = 1. \tag{7.2.13}$$

Since the problem (7.2.12) is posed on the whole line, it is natural to use a spectral method based on Hermite functions. Although the standard Hermite functions could be used as basis functions here, they are not the most appropriate ones. Below, we construct properly scaled Hermite functions which are eigenfunctions of B.

We recall that the standard Hermite polynomials $H_l(z)$ satisfy

$$H_l''(z) - 2zH_l'(z) + 2lH_l(z) = 0, \qquad z\in\mathbb{R}, \quad l\geqslant 0, \tag{7.2.14}$$

$$\int_{-\infty}^{\infty} H_l(z)H_n(z)e^{-z^2}\,\mathrm{d}z = \sqrt{\pi}\,2^l\,l!\,\delta_{ln}, \qquad l,n\geqslant 0. \tag{7.2.15}$$

We define the scaled Hermite function

$$h_l(z) = e^{-\gamma_z z^2/2}\, H_l\left(\sqrt{\gamma_z} z\right) / \left(\sqrt{2^l\, l!}(\pi/\gamma_z)^{1/4}\right), \qquad z \in \mathbb{R}. \tag{7.2.16}$$

Substituting (7.2.16) into (7.2.14) and (7.2.15), we find that

$$-\frac{1}{2}h_l''(z) + \frac{\gamma_z^2 z^2}{2} h_l(z) = \mu_l^z h_l(z),\ z \in \mathbb{R}, \quad \mu_l^z = \frac{2l+1}{2}\gamma_z,\ l \geqslant 0, \tag{7.2.17}$$

$$\int_{-\infty}^{\infty} h_l(z) h_n(z) \mathrm{d}z = \int_{-\infty}^{\infty} \frac{1}{\sqrt{\pi 2^l l! 2^n n!}} H_l(z) H_n(z) e^{-z^2} \mathrm{d}z = \delta_{ln}, l, n \geqslant 0. \tag{7.2.18}$$

Hence, $\{h_l\}$ are the eigenfunctions of B defined in (7.2.12).

Now, let us define $X_N = \mathrm{span}\{h_l : l = 0, 1, \cdots, N\}$. The Hermite-spectral method for (7.2.12) is to find $\psi_N(z,t) \in X_N$, i.e.,

$$\psi_N(z,t) = \sum_{l=0}^{N} \hat{\psi}_l(t)\ h_l(z), \qquad z \in \mathbb{R}, \tag{7.2.19}$$

such that

$$i\,\frac{\partial \psi_N(z,t)}{\partial t} = B\psi_N(z,t) = -\frac{1}{2}\frac{\partial^2 \psi_N(z,t)}{\partial z^2} + \frac{\gamma_z^2 z^2}{2}\psi_N(z,t),\ z \in \mathbb{R}. \tag{7.2.20}$$

Note that $\lim_{|z|\to+\infty} h_l(z) = 0$ (cf. [155]) so the decaying condition $\lim_{|z|\to+\infty} \psi_N(z,t) = 0$ is automatically satisfied.

Plugging (7.2.19) into (7.2.20), thanks to (7.2.17) and (7.2.18), we find

$$i\,\frac{d\hat{\psi}_l(t)}{dt} = \mu_l^z\ \hat{\psi}_l(t) = \frac{2l+1}{2}\gamma_z\ \hat{\psi}_l(t), \qquad l = 0, 1, \cdots, N. \tag{7.2.21}$$

Hence, the solution for (7.2.20) is given by

$$\psi_N(z,t) = e^{-iB(t-t_s)}\psi_N(z,t_s) = \sum_{l=0}^{N} e^{-i\mu_l^z (t-t_s)}\ \hat{\psi}_l(t_s) h_l(z),\ t \geqslant t_s. \tag{7.2.22}$$

As is standard in all spectral algorithms, a practical implementation will involve a Gauss-type quadrature. In this case, we need to use Gauss quadrature points and weights associated with the scaled Hermite functions.

Let $\{\hat{z}_k, \hat{\omega}_k^z\}_{k=0}^N$ be the Hermite-Gauss points and weights, i.e., $\{x_k, \omega_k^z\}_{k=0}^N$ de-

scribed in Theorem 4.1.1. Then, we have from (4.1.8) and (4.1.2) that

$$\sum_{k=0}^{N} \hat{\omega}_k^z \, \frac{H_l(\hat{z}_k)}{\pi^{1/4}\sqrt{2^l\, l!}} \, \frac{H_n(\hat{z}_k)}{\pi^{1/4}\sqrt{2^n\, n!}} = \delta_{ln}, \qquad l, n = 0, 1, \cdots, N. \tag{7.2.23}$$

Now let us define the scaled Hermite-Gauss points and weights by

$$z_k = \hat{z}_k/\sqrt{\gamma_z}, \qquad \omega_k^z = \hat{\omega}_k^z \, e^{\hat{z}_k^2}/\sqrt{\gamma_z}, \qquad 0 \leqslant k \leqslant N. \tag{7.2.24}$$

We then derive from (7.2.16) that

$$\begin{aligned} \sum_{k=0}^{N} \omega_k^z \, h_l(z_k) \, h_m(z_k) &= \sum_{k=0}^{N} \hat{\omega}_k^z \, e^{\hat{z}_k^2}/\sqrt{\gamma_z} \, h_l\left(\hat{z}_k/\sqrt{\gamma_z}\right) \, h_m\left(\hat{z}_k/\sqrt{\gamma_z}\right) \\ &= \sum_{k=0}^{N} \hat{\omega}_k^z \, \frac{H_l(\hat{z}_k)}{\pi^{1/4}\sqrt{2^l\, l!}} \, \frac{H_n(\hat{z}_k)}{\pi^{1/4}\sqrt{2^n\, n!}} = \delta_{ln}, \quad 0 \leqslant l, n \leqslant N. \end{aligned} \tag{7.2.25}$$

We are now ready to describe the full algorithm. Let ψ_k^n be the approximation of $\psi(z_k, t_n)$ and ψ^n be the solution vector with components ψ_k^n.

The fourth-order time-splitting Hermite-spectral method for 1-D GPE (7.2.3) is given by

$$\begin{aligned} \psi_k^{(1)} &= e^{-i2w_1 \, \Delta t \, \beta_1 |\psi_k^n|^2} \, \psi_k^n, & \psi_k^{(2)} &= \mathcal{F}_h(w_2, \psi^{(1)})_k, \\ \psi_k^{(3)} &= e^{-i2w_3 \, \Delta t \, \beta_1 |\psi_k^{(2)}|^2} \, \psi_k^{(2)}, & \psi_k^{(4)} &= \mathcal{F}_h(w_4, \psi^{(3)})_k, \\ \psi_k^{(5)} &= e^{-i2w_3 \, \Delta t \, \beta_1 |\psi_k^{(4)}|^2} \, \psi_k^{(4)}, & \psi_k^{(6)} &= \mathcal{F}_h(w_2, \psi^{(5)})_k, \\ \psi_k^{n+1} &= e^{-i2w_1 \, \Delta t \, \beta_1 |\psi_k^{(6)}|^2} \, \psi_k^{(6)}, & k &= 0, 1, \cdots, N, \end{aligned} \tag{7.2.26}$$

where $w_i, \ i = 1, 2, 3, 4$ are given in (1.6.30), and $\mathcal{F}_h(w, U)_k$ $(0 \leqslant k \leqslant N)$ can be computed from any given $w \in \mathbb{R}$ and $U = (U_0, \cdots, U_N)^{\mathrm{T}}$:

$$\mathcal{F}_h(w, U)_k = \sum_{l=0}^{N} e^{-i2w \, \mu_l^z \, \Delta t} \, \widehat{U}_l \, h_l(z_k), \quad \widehat{U}_l = \sum_{k=0}^{N} \omega_k^z \, U(z_k) \, h_l(z_k). \tag{7.2.27}$$

The memory requirement of this scheme is $\mathcal{O}(N)$ and the computational cost per time step is a small multiple of N^2 which comes from the evaluation of inner products in (7.2.27). Since each of the sub-problems (7.2.8) and (7.2.9) is conservative and our

numerical scheme solves the two sub-problems exactly in the discrete space, one can easily establish the following result (cf. [10]):

Lemma 7.2.1 *The time-splitting Hermite-spectral method* $(7.2.26)$ *is conservative,* i.e.,

$$\|\psi^n\|_{l^2}^2 = \sum_{k=0}^{N} \omega_k^z |\psi_k^n|^2 = \sum_{k=0}^{N} \omega_k^z |\psi_0(z_k)|^2 = \|\psi_0\|_{l^2}^2, \quad n = 0, 1, \cdots, \quad (7.2.28)$$

where

$$\|\psi\|_{l^2}^2 := \sum_{k=0}^{N} \omega_k^z |\psi(z_k)|^2. \quad (7.2.29)$$

Laguerre-spectral method for 2-D GPE with radial symmetry

In the 2-D case with radial symmetry, i.e. $d = 2$ and $\gamma_x = \gamma_y$ in (7.2.3), and $\psi_0(x, y) = \psi_0(r)$ in (7.2.3) with $r = \sqrt{x^2 + y^2}$, we can write the solution of (7.2.3) as $\psi(x, y, t) = \psi(r, t)$. Therefore, equation (7.2.9) collapses to

$$\begin{aligned} & i\,\frac{\partial \psi(r,t)}{\partial t} = B\psi(r,t) = -\frac{1}{2r}\frac{\partial}{\partial r}\left(r\frac{\partial \psi(r,t)}{\partial r}\right) + \frac{\gamma_r^2 r^2}{2}\psi(r,t), \quad 0 < r < \infty, \\ & \lim_{r\to\infty} \psi(r,t) = 0, \qquad t \geqslant 0, \end{aligned} \quad (7.2.30)$$

where $\gamma_r = \gamma_x = \gamma_y$. The normalization (7.2.5) reduces to

$$\|\psi(\cdot, t)\|^2 = 2\pi \int_0^\infty |\psi(r,t)|^2 r \, \mathrm{d}r \equiv 2\pi \int_0^\infty |\psi_0(r)|^2 r \, \mathrm{d}r = 1. \quad (7.2.31)$$

Note that it can be shown, similarly as for the Poisson equation in a 2-D disk (cf. [142]), that the problem (7.2.30) admits a unique solution without any condition at the pole $r = 0$.

Since (7.2.30) is posed on a semi-infinite interval, it is natural to consider Laguerre functions which have been successfully used for other problems in semi-infinite intervals (cf. [52], [143]). Again, the standard Laguerre functions, although usable, are not the most appropriate for this problem. Below, we construct properly scaled Laguerre functions which are eigenfunctions of B.

Let $\hat{L}_m(r)$ $(m = 0, 1, \cdots, M)$ be the Laguerre polynomials of degree m satisfying

$$\begin{aligned} &r\hat{L}_m''(r)+(1-r)\hat{L}_m'(r)+m\hat{L}_m(r)=0, \qquad m=0,1,\cdots,\\ &\int_0^\infty e^{-r}\ \hat{L}_m(r)\ \hat{L}_n(r)\ \mathrm{d}r=\delta_{mn}, \qquad m,n=0,1,\cdots. \end{aligned} \tag{7.2.32}$$

We define the scaled Laguerre functions L_m by

$$L_m(r)=\sqrt{\frac{\gamma_r}{\pi}}\ e^{-\gamma_r r^2/2}\ \hat{L}_m(\gamma_r r^2), \qquad 0\leqslant r<\infty. \tag{7.2.33}$$

Note that $\lim_{|r|\to+\infty} L_m(r)=0$ (cf. [155]) hence, $\lim_{|r|\to+\infty}\psi_M(r,t)=0$ is automatically satisfied.

Substituting (7.2.33) into (7.2.32), a simple computation shows

$$\begin{aligned} &-\frac{1}{2r}\frac{\partial}{\partial r}\left(r\frac{\partial L_m(r)}{\partial r}\right)+\frac{1}{2}\gamma_r^2 r^2 L_m(r)=\mu_m^r L_m(r),\ \mu_m^r=\gamma_r(2m+1),\ m\geqslant 0,\\ &2\pi\int_0^\infty L_m(r)L_n(r)r\ \mathrm{d}r=\int_0^\infty e^{-r}\hat{L}_m(r)\hat{L}_n(r)\ \mathrm{d}r=\delta_{mn}, \qquad m,n\geqslant 0. \end{aligned} \tag{7.2.34}$$

Hence, $\{L_m\}$ are the eigenfunctions of B defined in (7.2.30).

Let $Y_M=\mathrm{span}\{L_m:\ m=0,1,\cdots,M\}$. The Laguerre-spectral method for (7.2.30) is to find $\psi_M(r,t)\in Y_M$, i.e.,

$$\psi_M(r,t)=\sum_{m=0}^{M}\hat{\psi}_m(t)\ L_m(r), \qquad 0\leqslant r<\infty, \tag{7.2.35}$$

such that

$$i\ \frac{\partial\psi_M(r,t)}{\partial t}=B\psi_M(r,t)=-\frac{1}{2r}\frac{\partial}{\partial r}\left(r\frac{\partial\psi_M(r,t)}{\partial r}\right)+\frac{\gamma_r^2 r^2}{2}\psi_M(r,t), \quad 0<r<\infty. \tag{7.2.36}$$

Plugging (7.2.35) into (7.2.36), we find, thanks to (7.2.34),

$$i\ \frac{d\hat{\psi}_m(t)}{dt}=\mu_m^r\hat{\psi}_m(t)=\gamma_z(2m+1)\hat{\psi}_m(t), \quad m=0,1,\cdots,M. \tag{7.2.37}$$

Hence, the solution for (7.2.36) is given by

$$\psi_M(r,t)=e^{-iB(t-t_s)}\psi_M(r,t_s)=\sum_{m=0}^{M} e^{-i\mu_m^r(t-t_s)}\hat{\psi}_m(t_s)L_m(r), t\geqslant t_s. \quad (7.2.38)$$

We now derive the Gauss-Radau points and weights associated with the scaled Laguerre functions. Let $\{\hat{r}_j,\hat{\omega}_j^r\}_{j=0}^M$ be the Laguerre-Gauss-Radau points and weights, i.e., $\{x_j^{(0)},\omega_j^{(0)}\}$ given in (4.2.8). We have from (4.2.9) and (4.2.2) that

$$\sum_{j=0}^{M}\hat{\omega}_j^r\hat{L}_m(\hat{r}_j)\hat{L}_n(\hat{r}_j)=\delta_{nm},\qquad n,m=0,1,\cdots,M.$$

We define the scaled Laguerre-Gauss-Radau points r_j and weights ω_j^z by

$$\omega_j^r=\pi\hat{\omega}_j^r\ e^{\hat{r}_j}/\gamma_r,\qquad r_j=\sqrt{\hat{r}_j/\gamma_r},\qquad j=0,1,\cdots,M. \quad (7.2.39)$$

Hence, we have from (7.2.33) that

$$\begin{aligned}\sum_{j=0}^{M}\omega_j^r L_m(r_j)L_n(r_j)&=\sum_{j=0}^{M}\hat{\omega}_j^r e^{\hat{r}_j}\pi/\gamma_r\ L_m(\sqrt{\hat{r}_j/\gamma_r})\ L_n(\sqrt{\hat{r}_j/\gamma_r})\\&=\sum_{j=0}^{M}\hat{\omega}_j^r\hat{L}_m(\hat{r}_j)\hat{L}_n(\hat{r}_j)=\delta_{nm},\quad n,m=0,1,\cdots,M.\end{aligned} \quad (7.2.40)$$

The time-splitting Laguerre-spectral method can now be described as follows: Let ψ_j^n be the approximation of $\psi(r_j,t_n)$ and ψ^n be the solution vector with components ψ_j^n. Then, the fourth-order time-splitting Laguerre-pseudospectral (TSLP4) method for 2-D GPE (7.2.3) with radial symmetry is similar to (7.2.26) except that one needs to replace β_1 by β_2, N by M, the index k by j, and the operator $\mathcal{F}_h$ by $\mathcal{F}_L$ which is defined as

$$\mathcal{F}_L(w,U)_j=\sum_{l=0}^{M}e^{-i2w\ \mu_l^r\ \Delta t}\ \widehat{U}_l\ L_l(r_j),\quad \widehat{U}_l=\sum_{j=0}^{M}\omega_j^r\ U(r_j)\ L_l(r_j). \quad (7.2.41)$$

Similarly as in the Hermite case, the memory requirement of this scheme is $\mathcal{O}(M)$ and the computational cost per time step is a small multiple of M^2. As for the stability, we have

Lemma 7.2.2 *The time-splitting Laguerre-pseudospectral (TSLP4) method is conservative,* i.e.,

$$\|\psi^n\|_{l^2}^2 = \sum_{j=0}^{M} \omega_j^r |\psi_j^n|^2 = \sum_{j=0}^{M} \omega_j^r |\psi_0(r_j)|^2 = \|\psi_0\|_{l^2}^2, \qquad n \geqslant 0.$$

Laguerre-Hermite pseudospectral method for 3-D GPE with cylindrical symmetry

In the 3-D case with cylindrical symmetry, i.e., $d = 3$ and $\gamma_x = \gamma_y$ in (7.2.3), and $\psi_0(x, y, z) = \psi_0(r, z)$ in (7.2.3), the solution of (7.2.3) with $d = 3$ satisfies $\psi(x, y, z, t) = \psi(r, z, t)$. Therefore, Eq. (7.2.9) becomes

$$\begin{aligned} i\,\frac{\partial\psi(r,z,t)}{\partial t} &= B\psi(r,z,t) = -\frac{1}{2}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial\psi}{\partial r}\right)+\frac{\partial^2\psi}{\partial z^2}\right]+\frac{1}{2}\left(\gamma_r^2 r^2+\gamma_z^2 z^2\right)\psi, \\ & 0 < r < \infty,\ -\infty < z < \infty, \\ \lim_{r\to\infty}\psi(r,z,t) &= 0, \quad \lim_{|z|\to\infty}\psi(r,z,t) = 0, \qquad t \geqslant 0, \end{aligned} \tag{7.2.42}$$

where $\gamma_r = \gamma_x = \gamma_y$. The normalization (7.2.5) now is

$$\|\psi(\cdot,t)\|^2 = 2\pi\int_0^\infty\int_{-\infty}^\infty |\psi(r,z,t)|^2 r\,\mathrm{d}z\mathrm{d}r \equiv \|\psi_0\|^2 = 1. \tag{7.2.43}$$

Since the two-dimensional computational domain here is a tensor product of a semi-infinite interval and the whole line, it is natural to combine the Hermite-spectral and Laguerre-spectral methods. In particular, the product of scaled Hermite and Laguerre functions $\{L_m(r)h_l(z)\}$ are eigenfunctions of B defined in (7.2.42), since we derive from (7.2.17) and (7.2.34) that

$$\begin{aligned} & -\frac{1}{2}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial}{\partial r}\right)+\frac{\partial^2}{\partial z^2}\right](L_m(r)\,h_l(z))+\frac{1}{2}\left(\gamma_r^2 r^2+\gamma_z^2 z^2\right)(L_m(r)\,h_l(z)) \\ &= \left[-\frac{1}{2r}\frac{d}{dr}\left(r\frac{dL_m(r)}{dr}\right)+\frac{1}{2}\gamma_r^2 r^2 L_m(r)\right]h_l(z) \\ &\quad + \left[-\frac{1}{2}\frac{d^2h_l(z)}{dz^2}+\frac{1}{2}\gamma_z^2 z^2 h_l(z)\right]L_m(r) \\ &= \mu_m^r L_m(r)h_l(z) + \mu_l^z h_l(z)L_m(r) = (\mu_m^r+\mu_l^z)L_m(r)h_l(z). \end{aligned} \tag{7.2.44}$$

Now, let

$$X_{MN} = \text{span}\{L_m(r)h_l(z) : \ m = 0, 1, \cdots, M, \ l = 0, 1, \cdots, N\}.$$

The Laguerre-Hermite spectral method for (7.2.42) is to find $\psi_{MN}(r, z, t) \in X_{MN}$, i.e.,

$$\psi_{MN}(r, z, t) = \sum_{m=0}^{M} \sum_{l=0}^{N} \tilde{\psi}_{ml}(t) \ L_m(r) \ h_l(z), \tag{7.2.45}$$

such that

$$\begin{aligned} i \ \frac{\partial \psi_{MN}(r, z, t)}{\partial t} &= B\psi_{MN}(r, z, t) \\ &= -\frac{1}{2}\left[\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial \psi_{MN}}{\partial r}\right) + \frac{\partial^2 \psi_{MN}}{\partial z^2}\right] + \frac{1}{2}\left(\gamma_r^2 r^2 + \gamma_z^2 z^2\right)\psi_{MN}. \end{aligned} \tag{7.2.46}$$

Using (7.2.45) into (7.2.46), we find, thanks to (7.2.44),

$$i \ \frac{d\tilde{\psi}_{ml}(t)}{dt} = (\mu_m^r + \mu_l^z)\,\tilde{\psi}_{ml}(t), \quad m = 0, 1, \cdots, M, \ l = 0, 1, \cdots, N. \tag{7.2.47}$$

Hence, the solution for (7.2.46) is given by

$$\begin{aligned} \psi_{MN}(r, z, t) &= e^{-iB(t-t_s)}\psi_{MN}(r, z, t_s) \\ &= \sum_{m=0}^{M} \sum_{l=0}^{N} e^{-i(\mu_m^r + \mu_l^z)(t-t_s)} \tilde{\psi}_{ml}(t_s) L_m(r) \ h_l(z), \qquad t \geqslant t_s. \end{aligned} \tag{7.2.48}$$

In summary, let ψ_{jk}^n be the approximation of $\psi(r_j, z_k, t_n)$ and ψ^n the solution vector with components ψ_{jk}^n. The fourth-order time-splitting Laguerre-Hermite-pseudospectral method for the 3-D GPE (7.2.3) with cylindrical symmetry is essentially the same as (7.2.26), except that now we replace β_1 by β_3, the index k ($0 \leqslant k \leqslant N$) by jk ($0 \leqslant j \leqslant M$, $0 \leqslant k \leqslant N$), and the operator $\mathcal{F}_h$ by $\mathcal{F}_{Lh}$ defined by

$$\begin{aligned} \mathcal{F}_{Lh}(w, U)_{jk} &= \sum_{m=0}^{M} \sum_{l=0}^{N} e^{-i2w\Delta t(\mu_m^r + \mu_l^z)} \ \widehat{U}_{ml} \ L_m(r_j) h_l(z_k), \\ \widehat{U}_{ml} &= \sum_{j=0}^{M} \sum_{k=0}^{N} \omega_j^r \ \omega_k^z \ U(r_j, z_k) \ L_m(r_j) h_l(z_k). \end{aligned} \tag{7.2.49}$$

The memory requirement of this scheme is $\mathcal{O}(MN)$ and the computational cost per time step is $\mathcal{O}(\max(M^2N, N^2M))$. Obviously, we have

Lemma 7.2.3 *The time-splitting Laguerre-Hermite pseudospectral method is conservative in the sense that*

$$\begin{aligned}\|\psi^n\|_{l^2}^2 &= \sum_{j=0}^{M}\sum_{k=0}^{N}\omega_j^r\omega_k^z|\psi_{jk}^n|^2 \\ &= \sum_{j=0}^{M}\sum_{k=0}^{N}\omega_j^r\omega_k^z|\psi_0(r_j,z_k)|^2 = \|\psi_0\|_{l^2}^2, \quad n\geqslant 0.\end{aligned} \tag{7.2.50}$$

Numerical results

We now present some numerical results. We define the condensate width along the r- and z-axis as

$$\sigma_\alpha^2 = \int_{\mathbb{R}^d}\alpha^2|\psi(\mathbf{x},t)|\,\mathrm{d}\mathbf{x}, \quad \alpha = x,y,z, \qquad \sigma_r^2 = \sigma_x^2+\sigma_y^2.$$

Example 7.2.1 *The 1-D Gross-Pitaevskii equation: We choose* $d=1$, $\gamma_z=2$, $\beta_1=50$ *in* (7.2.3)*. The initial data* $\psi_0(z)$ *is chosen as the ground state of the 1-D GPE* (7.2.3) *with* $d=1$, $\gamma_z=1$ *and* $\beta_1=50$*. This corresponds to an experimental setup where initially the condensate is assumed to be in its ground state, and the trap frequency is doubled at* $t=0$*.*

We solve this problem by using (7.2.26) with $N=31$ and time step $k=0.001$. Figure 7.5 plots the condensate width and central density $|\psi(0,t)|^2$ as functions of

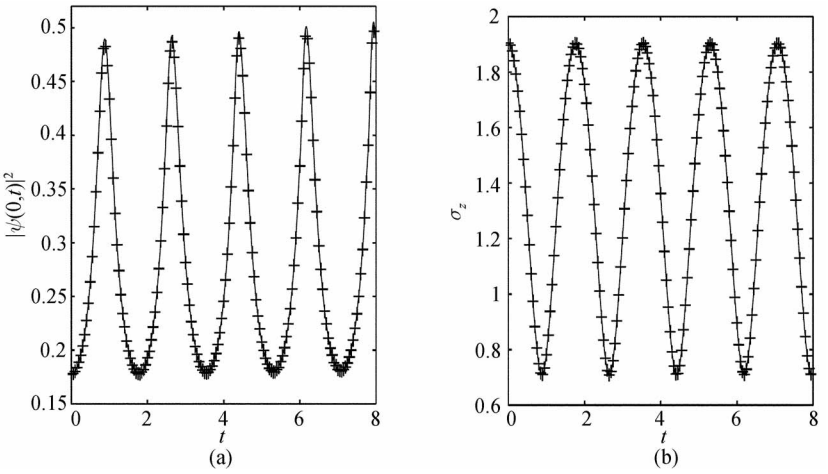

Figure 7.5 Evolution of central density and condensate width in Example 7.2.1. '—': 'exact solutions' obtained by the TSSP [8] with 513 grid points over an interval $[-12,12]$; '+ + + ': Numerical results by (7.2.26) with 31 grid points on the whole z-axis.
(a) Central density $|\psi(0,t)|^2$; (b) condensate width σ_z.

time. Our numerical experiments also show that the scheme (7.2.26) with $N = 31$ gives similar numerical results as the TSSP method [8] for this example, with 513 grid points over the interval $[-12, 12]$ and time step $k = 0.001$.

In order to test the 4th-order accuracy in time of (7.2.26), we compute a numerical solution with a very fine mesh, e.g. $N = 81$, and a very small time step, e.g. $\Delta t = 0.0001$, as the 'exact' solution ψ. Let $\psi^{\Delta t}$ denote the numerical solution under $N = 81$ and time step Δt. Since N is large enough, the truncation error from space discretization is negligible compared to that from time discretization. Table 7.3 shows the errors $\max |\psi(t) - \psi^{\Delta t}(t)|$ and $\|\psi(t) - \psi^{\Delta t}(t)\|_{l^2}$ at $t = 2.0$ for different time steps Δt. The results in Table 7.3 demonstrate the 4th-order accuracy in time of (7.2.26).

Table 7.3 Time discretization error analysis for (7.2.26) (At $t = 2.0$ with $N = 81$)

Δt	$1/40$	$1/80$	$1/160$	$1/320$
$\max \|\psi(t) - \psi^{\Delta t}(t)\|$	0.1619	4.715E-6	3.180E-7	2.036E-8
$\|\psi(t) - \psi^{\Delta t}(t)\|_{l^2}$	0.2289	7.379E-6	4.925E-7	3.215E-8

Example 7.2.2 *The 2-D Gross-Pitaevskii equation with radial symmetry: we choose* $d = 2$, $\gamma_r = \gamma_x = \gamma_y = 2$, $\beta_2 = 50$ *in* (7.2.3). *The initial data* $\psi_0(r)$ *is chosen as the ground state of the 2-D GPE* (7.2.3) *with* $d = 2$, $\gamma_r = \gamma_x = \gamma_y = 1$ *and* $\beta_2 = 50$. *Again this corresponds to an experimental setup where initially the condensate is assumed to be in its ground state, and the trap frequency is doubled at* $t = 0$.

We solve this problem by using the time splitting Laguerre-spectral method with $M = 30$ and time step $k = 0.001$. Figure 7.6 plots the condensate width and central

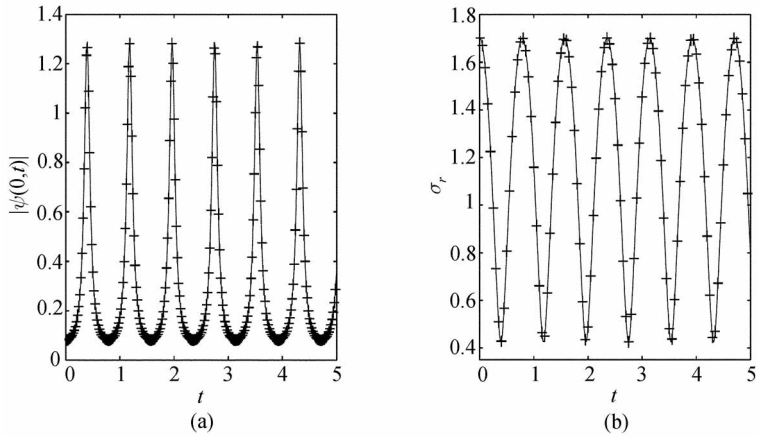

Figure 7.6 Evolution of central density and condensate width in Example 7.2.2. '—': 'exact solutions' obtained by TSSP [8] with 513^2 grid points over a box $[-8, 8]^2$; '+': Numerical results by our scheme with 30 grid points on the semi-infinite interval $[0, \infty)$. (a) Central density $|\psi(0, t)|^2$; (b) condensate width σ_r.

density $|\psi(0,t)|^2$ as functions of time. Our numerical experiments also show that our scheme with $M = 30$ gives similar numerical results as the TSSP method [8] for this example, with 513^2 grid points over the box $[-8,8]^2$ and time step $k = 0.001$.

Exercise 7.2

Problem 1 Prove Lemma 7.2.1.

7.3 Spectral approximation of the Stokes equations

Spectral-Galerkin method for the Stokes problem
A simple iterative method – the Uzawa algorithm
Error analysis

The Stokes equations play an important role in fluid mechanics and solid mechanics. Numerical approximation of Stokes equations has attracted considerable attention in the last few decades and is still an active research direction (cf. [55], [24], [11] and the references therein).

We consider the Stokes equations in primitive variables:

$$\begin{cases} -\nu\Delta \boldsymbol{u} + \nabla p = \boldsymbol{f}, & \text{in } \Omega \subset R^d, \\ \nabla\cdot\boldsymbol{u} = 0, \quad \text{in } \Omega; & \boldsymbol{u}|_{\partial\Omega} = 0. \end{cases} \tag{7.3.1}$$

In the above, the unknowns are the velocity vector $\boldsymbol{u}$ and the pressure p; $\boldsymbol{f}$ is a given body force and ν is the viscosity coefficient. For the sake of simplicity, the homogeneous Dirichlet boundary condition is assumed, although other admissible boundary conditions can be treated similarly (cf. [55]).

Let us denote by $A : H_0^1(\Omega)^d \to H^{-1}(\Omega)^d$ the Laplace operator defined by

$$\langle A\boldsymbol{u}, \boldsymbol{v}\rangle_{H^{-1},H_0^1} = (\nabla\boldsymbol{u}, \nabla\boldsymbol{v}), \quad \forall \boldsymbol{u}, \boldsymbol{v} \in H_0^1(\Omega)^d. \tag{7.3.2}$$

Then, applying the operator $\nabla\cdot A^{-1}$ to (7.3.1), we find that the pressure can be determined by

$$Bp := -\nabla\cdot A^{-1}\nabla p = -\nabla\cdot A^{-1}\boldsymbol{f}. \tag{7.3.3}$$

Once p is obtained from (7.3.3), we can obtain $\boldsymbol{u}$ from (7.3.1) by inverting the Laplace operator, namely,

$$\boldsymbol{u} = \frac{1}{\nu}A^{-1}(\boldsymbol{f} - \nabla p). \tag{7.3.4}$$

Let $L_0^2(\Omega) = \{q \in L^2(\Omega) : \int_\Omega q\mathrm{d}x = 0\}$. The operator $B := -\nabla\cdot A^{-1}\nabla : \ L_0^2(\Omega) \to L_0^2(\Omega)$ is usually referred to as Uzawa operator or the Schur complement associated

with the Stokes operator. We have

$$(Bp, q) := -(\nabla\cdot A^{-1}\nabla p, q) = (A^{-1}\nabla p, \nabla q) = (p, Bq). \tag{7.3.5}$$

Therefore, B is a "zero-th order" self-adjoint positive definite operator, and we can expect that the corresponding discrete operator can be inverted efficiently by using a suitable iterative method such as the conjugate gradient method.

Spectral-Galerkin method for the Stokes problem

To simplify the presentation, we shall consider only $\Omega = (-1, 1)^d$ with $d = 2$ or 3. Let $\boldsymbol{X}_N$ and M_N be a suitable pair of finite dimensional approximate spaces for $H_0^1(\Omega)^d$ and $L_0^2(\Omega)$. The corresponding Galerkin method for the Stokes problem is: Find $(\boldsymbol{u}_N, p_N) \in \boldsymbol{X}_N \times M_N$ such that

$$\begin{aligned} &\nu(\nabla\boldsymbol{u}_N, \nabla\boldsymbol{v}_N) - (p_N, \nabla\cdot\boldsymbol{v}_N) = (f, \boldsymbol{v}_N), \quad \forall\boldsymbol{v}_N \in \boldsymbol{X}_N, \\ &(\nabla\cdot\boldsymbol{u}_N, q_N) = 0, \quad \forall q_N \in M_N. \end{aligned} \tag{7.3.6}$$

It is well-known (see [55]) that the discrete problem (7.3.6) admits a unique solution if and only if there exists a positive constant β_N such that

$$\inf_{q_N\in M_N}\sup_{\boldsymbol{v}_N\in\boldsymbol{X}_N}\frac{(q_N, \nabla\cdot\boldsymbol{v}_N)}{\|q_N\|_0\|\nabla\boldsymbol{v}_N\|_0} \geqslant \beta_N. \tag{7.3.7}$$

The above condition is referred as Brezzi-Babuska inf-sup condition (cf. [7], [23]) and β_N is referred as the inf-sup constant.

Let $\{\phi_k\}_{k=1}^{N_u}$ and $\{\psi_k\}_{k=1}^{N_p}$ be respectively basis functions for $\boldsymbol{X}_N$ and M_N. Then we can write

$$\boldsymbol{u}_N = \sum_{k=1}^{N_u}\tilde{u}_k\phi_k, \quad p_N = \sum_{k=1}^{N_p}\tilde{p}_k\psi_k. \tag{7.3.8}$$

Set

$$\begin{aligned} &a_{ij} = (\nabla\phi_j, \nabla\phi_i), && A_N = (a_{ij})_{i,j=1,\cdots,N_u}, \\ &b_{ij} = -(\psi_j, \nabla\cdot\phi_i), && B_N = (b_{ij})_{i=1,\cdots,N_u, j=1,\cdots,N_p}, \\ &\bar{u} = (\tilde{u}_1, \cdots, \tilde{u}_{N_u})^t, && \bar{p} = (\tilde{p}_1, \cdots, \tilde{p}_{N_p})^t, \\ &f_j = (I_N\boldsymbol{f}, \phi_j), && \bar{f} = (f_1, \cdots, f_{N_u})^{\mathrm{T}}. \end{aligned} \tag{7.3.9}$$

Then the matrix form of (7.3.6) is

$$\begin{aligned} \nu A_N\bar{u} + B_N^t\bar{p} &= \bar{f}, \\ B_N\bar{u} &= 0. \end{aligned} \tag{7.3.10}$$

As in the space continuous case, $\bar{p}$ can be obtained by inverting the discrete Uzawa operator:

$$B_N A_N^{-1} B_N^t \bar{p} = B_N A_N^{-1} \bar{f}. \tag{7.3.11}$$

It is easy to show that the discrete Uzawa operator is positive symmetric definite if and only if there exists $\beta_N > 0$ such that the inf-sup condition is satisfied; furthermore, it is shown (see [114]) that

$$\mathrm{cond}(B_N A_N^{-1} B_N^t) = \beta_N^{-2}. \tag{7.3.12}$$

Therefore, the effectiveness of the Uzawa algorithm is directly related to the size of β_N.

It is customary in a spectral approximation to take $\boldsymbol{X}_N = (P_N \cap H_0^1(\Omega))^d$. However, how to choose M_N is a non-trivial question. For any given M_N, let us define

$$Z_N = \{q_N \in M_N : (q_N, \nabla \cdot \boldsymbol{v}_N) = 0, \quad \forall \boldsymbol{v}_N \in \boldsymbol{X}_N\}. \tag{7.3.13}$$

Obviously if $(\boldsymbol{u}_N, p_N)$ is a solution of (7.3.6), then so is $(\boldsymbol{u}_N, p_N + q_N)$ for any $q_N \in Z_N$. Hence, any mode in Z_N is called a spurious mode. For the most obvious choice $M_N = \{q_N \in P_N : \int_\Omega q_N \mathrm{d}x = 0\}$, one can verify that Z_N spans a seven-dimensional space if $d = 2$ and $12N + 3$-dimensional space if $d = 3$. Therefore, it is not a good choice for the pressure space. On the other hand, if we set

$$M_N = \{q_N \in P_{N-2} : \int_\Omega q_N \mathrm{d}x = 0\},$$

then the corresponding Z_N is empty and this leads to a well-posed problem (7.3.6) with the inf-sup constant $\beta_N \sim N^{-(d-1)/2}$ (see [11]).

Remark 7.3.1 It is also shown in [12] that for any given $0 < \lambda < 1$, $M_N^{(\lambda)} = \{q_N \in P_{\lambda N} : \int_\Omega q_N \mathrm{d}x = 0\}$ leads to a well-posed problem (7.3.6) with an inf-sup constant which is independent of N but is of course dependent on λ in such a way that $\beta_N \to 0$ as $\lambda \to 1^-$.

A simple iterative method – the Uzawa algorithm

We now give a brief presentation of the Uzawa algorithm which was originated from the paper by Arrow, Hurwicz and Uzawa[4] had been used frequently in finite element approximations of the Stokes problem (see [162] and the references therein).

Given an arbitrary $p^0 \in L_0^2(\Omega)$, the Uzawa algorithm consists of defining $(\boldsymbol{u}^{k+1},$

p^{k+1}) recursively by

$$\begin{cases} -\nu\Delta \boldsymbol{u}^{k+1} + \nabla p^k = \boldsymbol{f}, \qquad \boldsymbol{u}|_{\partial\Omega} = 0; \\ p^{k+1} = p^k - \rho_k \nabla\cdot \boldsymbol{u}^{k+1}, \end{cases} \tag{7.3.14}$$

where ρ_k is a suitable positive sequence to be specified. By eliminating $\boldsymbol{u}^{k+1}$ from (7.3.14), we find that

$$p^{k+1} = p^k - \frac{\rho_k}{\nu}(Bp^k + \nabla \cdot A^{-1}\boldsymbol{f}). \tag{7.3.15}$$

Thus, the Uzawa algorithm (7.3.14) is simply a Richardson iteration for (7.3.3). We now investigate the convergence properties of the Uzawa algorithm.

It can be shown (cf. [120]) that there exists $0 < \beta \leqslant 1$ such that

$$\beta\|q\|^2 \leqslant (Bq, q) \leqslant \|q\|^2, \quad \forall q \in L_0^2(\Omega). \tag{7.3.16}$$

Let us denote $\alpha = \min\{\beta\min_k \rho_k/\nu, 2 - \max_k \rho_k/\nu\}$. Then, an immediate consequence of (7.3.16) is that $I - \rho_k B/\nu$ is a strict contraction in $L_0^2(\Omega)$. Indeed, we derive from (7.3.16) that

$$\left(1 - \frac{\max_k \rho_k}{\nu}\right)\|q\|^2 \leqslant \left(\left(I - \frac{\rho_k}{\nu}B\right)q, q\right) \leqslant \left(1 - \beta\frac{\min_k \rho_k}{\nu}\right)\|q\|^2$$

which implies that

$$\left|\left(\left(I - \frac{\rho_k}{\nu}B\right)q, q\right)\right| \leqslant (1-\alpha)\|q\|^2. \tag{7.3.17}$$

Remark 7.3.2 A particularly simple and effective choice is to take $\rho_k = \nu$. In this case, we have $\alpha = \beta$ and the following convergence result:

$$\|\boldsymbol{u}^k - \boldsymbol{u}\|_1 + \|p^k - p\| \lesssim (1-\beta)^k. \tag{7.3.18}$$

Remark 7.3.3 Consider the Legendre-Galerkin approximation of the Uzawa algorithm: Given an arbitrary p_N^0, define $(\boldsymbol{u}_N^{k+1}, p_N^{k+1}) \in \boldsymbol{X}_N \times M_N$ recursively from

$$\begin{aligned} &\nu(\nabla \boldsymbol{u}_N^{k+1}, \nabla \boldsymbol{v}_N) - (p_N^k, \nabla\cdot\boldsymbol{v}_N) = (f, \boldsymbol{v}_N), \quad \forall \boldsymbol{v}_N \in \boldsymbol{X}_N, \\ &(p_N^{k+1}, q_N) = (p_N^k - \rho_k \nabla\cdot\boldsymbol{u}_N^{k+1}, q_N), \quad \forall q_N \in M_N. \end{aligned} \tag{7.3.19}$$

Then, by using the same procedure as in the continuous case, it can be shown that for

$\rho_k = \nu$ we have

$$\|\boldsymbol{u}_N^k - \boldsymbol{u}_N\|_1 + \|p_N^k - p_N\| \lesssim (1-\beta_N^2)^k, \tag{7.3.20}$$

where $(\boldsymbol{u}_N, p_N)$ is the solution of (7.3.6) and β_N is the inf-sup condition defined in (7.3.7). Thus, for a given tolerance ϵ, the number of Uzawa steps needed is proportional to $\beta_N^{-2}\log\epsilon$ while the number of the CG steps needed for the same tolerance, thanks to (7.3.12) and Theorem 1.7.1, is proportional to $\beta_N^{-1}\log\epsilon$. Therefore, whenever possible, one should always use the CG method instead of Uzawa algorithm.

Error analysis

The inf-sup constant β_N not only plays an important role in the implementation of the approximation (7.3.6), it is also of paramount importance in its error analysis. Let us denote

$$\boldsymbol{V}_N = \{\boldsymbol{v}_N \in \boldsymbol{X}_N : (q_N, \nabla\cdot\boldsymbol{v}_N) = 0, \quad \forall q_N \in M_N\}. \tag{7.3.21}$$

Then, with respect to the error analysis, we have

Theorem 7.3.1 *Assuming* (7.3.7)*, the following error estimates hold:*

$$\begin{aligned} &\|\boldsymbol{u} - \boldsymbol{u}_N\|_1 \lesssim \inf_{\boldsymbol{v}_N\in\boldsymbol{V}_N} \|\boldsymbol{u} - \boldsymbol{v}_N\|_1, \\ &\beta_N\|p - p_N\|_0 \lesssim \inf_{\boldsymbol{v}_N\in\boldsymbol{V}_N} \|\boldsymbol{u} - \boldsymbol{v}_N\|_1 + \inf_{q_N\in M_N}\|p - q_N\|_0, \end{aligned} \tag{7.3.22}$$

where $(\boldsymbol{u}, p)$ *and* $(\boldsymbol{u}_N, p_N)$ *are respectively the solution of* (7.3.1) *and* (7.3.6)*.*

Proof Let us denote

$$\boldsymbol{V} = \{\boldsymbol{v} \in H_0^1(\Omega)^d : (q, \nabla\cdot\boldsymbol{v}) = 0, \quad \forall q \in L_0^2(\Omega)\}. \tag{7.3.23}$$

Then, by the definition of $\boldsymbol{V}$ and $\boldsymbol{V}_N$,

$$\begin{aligned} &\nu(\nabla\boldsymbol{u}, \nabla\boldsymbol{v}) = (\boldsymbol{f}, \boldsymbol{v}), && \forall \boldsymbol{v} \in \boldsymbol{V}, \\ &\nu(\nabla\boldsymbol{u}_N, \nabla\boldsymbol{v}_N) = (f, \boldsymbol{v}_N), && \forall \boldsymbol{v}_N \in \boldsymbol{V}_N. \end{aligned} \tag{7.3.24}$$

Since $\boldsymbol{V}_N \subset \boldsymbol{V}$, we have $\nu(\nabla(\boldsymbol{u} - \boldsymbol{u}_N), \boldsymbol{v}_N) = 0, \forall \boldsymbol{v}_N \in \boldsymbol{V}_N$. Hence,

$$\|\nabla(\boldsymbol{u} - \boldsymbol{u}_N)\|^2 = (\nabla(\boldsymbol{u} - \boldsymbol{u}_N), \nabla(\boldsymbol{u} - \boldsymbol{u}_N)) = \inf_{\boldsymbol{v}_N\in\boldsymbol{V}_N} (\nabla(\boldsymbol{u} - \boldsymbol{u}_N), \nabla(\boldsymbol{u} - \boldsymbol{v}_N)),$$

which implies immediately

$$\|\nabla(\boldsymbol{u}-\boldsymbol{u}_N)\| \leqslant \inf_{\boldsymbol{v}_N\in\boldsymbol{V}_N}\|\nabla(\boldsymbol{u}-\boldsymbol{v}_N)\|.$$

Next, we derive from (7.3.1)–(7.3.6) the identity

$$\nu(\nabla(\boldsymbol{u}-\boldsymbol{u}_N),\nabla\boldsymbol{v}_N)-(p-p_N,\nabla\cdot\boldsymbol{v}_N)=0,\quad \forall\boldsymbol{v}_N\in\boldsymbol{X}_N. \tag{7.3.25}$$

Hence, by using (7.3.7) and the above identity, we find that for any $q_N\in M_N$,

$$\begin{aligned}\beta_N\|q_N-p_N\| &\leqslant \sup_{\boldsymbol{v}_N\in\boldsymbol{X}_N}\frac{(q_N-p_N,\nabla\cdot\boldsymbol{v}_N)}{\|\nabla\boldsymbol{v}_N\|}\\ &= \sup_{\boldsymbol{v}_N\in\boldsymbol{X}_N}\frac{\nu(\nabla(\boldsymbol{u}-\boldsymbol{u}_N),\nabla\boldsymbol{v}_N)-(p-q_N,\nabla\cdot\boldsymbol{v}_N)}{\|\nabla\boldsymbol{v}_N\|}\end{aligned}$$

It follows from the identity $\|\nabla\boldsymbol{v}\|=\|\nabla\times\boldsymbol{v}\|+\|\nabla\cdot\boldsymbol{v}\|,\forall\boldsymbol{v}\in H_0^1(\Omega)^d$, and the Cauchy-Schwarz inequality that

$$\beta_N\|q_N-p_N\|\leqslant\nu\|\nabla(\boldsymbol{u}-\boldsymbol{u}_N)\|+\|p-q_N\|,\quad\forall q_N\in M_N.$$

Therefore,

$$\begin{aligned}\beta_N\|p-p_N\| &\leqslant \beta_N\inf_{q_N\in M_N}(\|p-q_N\|+\|q_N-p_N\|)\\ &\lesssim \|\nabla(\boldsymbol{u}-\boldsymbol{u}_N)\|+\inf_{q_N\in M_N}\|p-q_N\|\\ &\lesssim \inf_{\boldsymbol{v}_N\in\boldsymbol{V}_N}\|\boldsymbol{u}-\boldsymbol{v}_N\|_1+\inf_{q_N\in M_N}\|p-q_N\|.\end{aligned}$$

This completes the proof of this theorem. □

We note in particular that the pressure approximation cannot be optimal if β_N is dependent on N.

Exercise 7.3

Problem 1 Implement the Uzawa algorithm for solving the Stokes problem with $M_N=P_{N-2}\cap L_0^2(\Omega)$ and $M_N=P_{\lambda N}\cap L_0^2(\Omega)$ for $\lambda=0.7,0.8,0.9$. Explain your results.

Problem 2 Prove the statement (7.3.14).

Problem 3 Prove the statements (7.3.18) and (7.3.20).

7.4 Spectral-projection method for Navier-Stokes equations

A second-order rotational pressure-correction scheme
A second-order consistent splitting scheme
Full discretization

The incompressible Navier-Stokes equations are fundamental equations of fluid dynamics. Accurate numerical approximations of Navier-Stokes equations play an important role in many scientific applications. There have been an enormous amount of research work, and still growing, on mathematical and numerical analysis of the Navier-Stokes equations. We refer to the books[162, 89, 40, 56] for more details on the approximation of Navier-Stokes equations by finite elements, spectral and spectral element methods. In this section, we briefly describe two robust and accurate projection type schemes and the related full discretization schemes with a spectral-Galerkin discretization in space. We refer to [69] for an up-to-date review on the subject related to projection type schemes for the Navier-Stokes equations.

We now consider the numerical approximations of the unsteady Navier-Stokes equations:

$$\begin{cases} \boldsymbol{u}_t - \nu\Delta\boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p = \boldsymbol{f}, & \text{in } \Omega\times(0,T], \\ \nabla\cdot\boldsymbol{u} = 0, & \text{in } \Omega\times[0,T], \end{cases} \tag{7.4.1}$$

subject to appropriate initial and boundary conditions for $\boldsymbol{u}$. In the above, the unknowns are the velocity vector $\boldsymbol{u}$ and the pressure p; $\boldsymbol{f}$ is a given body force, ν is the kinematic viscosity, Ω is an open and bounded domain in $\mathbb{R}^d$ ($d = 2$ or 3 in practical situations), and $[0, T]$ is the time interval.

As for the Stokes equation, one of the main difficulties in approximating (7.4.1) is that the velocity and the pressure are coupled by the incompressibility constraint $\nabla\cdot\boldsymbol{u} = 0$. Although the Uzawa algorithm presented in the previous section is efficient for the steady Stokes problem, it is in general very costly to apply an Uzawa-type iteration at each time step. A popular and effective strategy is to use a fractional step scheme to decouple the computation of the pressure from that of the velocity. This approach was first introduced by Chorin [30] and Temam[161] in the late 60's, and its countless variants have played and are still playing a major role in computational fluid dynamics, especially for large three-dimensional numerical simulations. We refer to [69] for an up-to-date review on this subject.

Below, we present an efficient and accurate spectral-projection method for (7.4.1). The spatial variables will be discretized by the Legendre spectral-Galerkin method

described in previous chapters while two time discretization schemes will be described: the first is the rotational pressure-correction scheme (see [163], [70]), the second is the second-order consistent splitting scheme recently introduced by Guermond and Shen [68].

A second-order rotational pressure-correction scheme

Assuming $(\boldsymbol{u}^k, \boldsymbol{u}^{k-1}, p^k)$ are known, in the first substep, we look for $\tilde{\boldsymbol{u}}^{k+1}$ such that

$$\begin{cases} \dfrac{1}{2\delta t}(3\tilde{\boldsymbol{u}}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}) - \nu\Delta\tilde{\boldsymbol{u}}^{k+1} + \nabla p^k = \boldsymbol{g}(t_{k+1}), \\ \tilde{\boldsymbol{u}}^{k+1}|_{\partial\Omega} = 0, \end{cases} \tag{7.4.2}$$

where $\boldsymbol{g}(t_{k+1}) = \boldsymbol{f}(t_{k+1}) - (2(\boldsymbol{u}^k \cdot \nabla)\boldsymbol{u}^k - (\boldsymbol{u}^{k-1} \cdot \nabla)\boldsymbol{u}^{k-1})$. Then, in the second substep, we determine $(\boldsymbol{u}^{k+1}, \phi^{k+1})$ such that

$$\begin{cases} \dfrac{1}{2\delta t}(3\boldsymbol{u}^{k+1} - 3\tilde{\boldsymbol{u}}^{k+1}) + \nabla\phi^{k+1} = 0, \\ \nabla\cdot\boldsymbol{u}^{k+1} = 0, \\ \boldsymbol{u}^{k+1}\cdot\boldsymbol{n}|_{\partial\Omega} = 0. \end{cases} \tag{7.4.3}$$

The remaining task is to define a suitable p^{k+1} so that we can advance to the next time step. To this end, we first notice from (7.4.3) that

$$\Delta\tilde{\boldsymbol{u}}^{k+1} = \Delta\boldsymbol{u}^{k+1} + \frac{2\delta t}{3}\nabla\Delta\phi^{k+1} = \Delta\boldsymbol{u}^{k+1} + \nabla\nabla\cdot\tilde{\boldsymbol{u}}^{k+1}.$$

We then sum up the two substeps and use the above identity to obtain:

$$\begin{cases} \dfrac{1}{2\delta t}(3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}) - \nu\Delta\boldsymbol{u}^{k+1} + \nabla(\phi^{k+1} + p^k - \nu\nabla\cdot\tilde{\boldsymbol{u}}^{k+1}) = \boldsymbol{g}(t_{k+1}), \\ \nabla\cdot\boldsymbol{u}^{k+1} = 0, \\ \boldsymbol{u}^{k+1}\cdot\boldsymbol{n}|_{\partial\Omega} = 0. \end{cases} \tag{7.4.4}$$

Therefore, it is clear that we should set

$$p^{k+1} = \phi^{k+1} + p^k - \nu\nabla\cdot\tilde{\boldsymbol{u}}^{k+1}. \tag{7.4.5}$$

We note that the only difference between (7.4.4) and (7.4.5) and a *coupled* second-

order scheme is that

$$\boldsymbol{u}^{k+1}\cdot\boldsymbol{\tau}\Big|_{\partial\Omega}=-\frac{2\delta t}{3}\nabla\phi^{k+1}\cdot\boldsymbol{\tau}\Big|_{\partial\Omega}\neq 0$$

(where τ is the tangential direction) but "small". Hence, it is expected that the scheme (7.4.2), (7.4.3) and (7.4.5) provides a good approximation to the Navier-Stokes equations. Indeed, it is shown in [71] that

$$\|\boldsymbol{u}(t_k)-\boldsymbol{u}^k\|+\sqrt{\delta t}(\|\boldsymbol{u}(t_k)-\boldsymbol{u}^k\|_1+\|p(t_k)-p^k\|)\lesssim\delta t^2. \tag{7.4.6}$$

In practice, the coupled system (7.4.3) is decoupled by taking the divergence of the first equation in (7.4.3), leading to:

$$\begin{aligned}&\Delta\phi^{k+1}=\frac{3}{2\delta t}\nabla\cdot\tilde{\boldsymbol{u}}^{k+1},\quad \frac{\partial\phi^{k+1}}{\partial\boldsymbol{n}}\Big|_{\partial\Omega}=0;\\ &\boldsymbol{u}^{k+1}=\tilde{\boldsymbol{u}}^{k+1}-\frac{2\delta t}{3}\nabla\phi^{k+1}.\end{aligned} \tag{7.4.7}$$

Hence, at each time step, the scheme (7.4.2)– (7.4.5) only involves inverting a Poisson-type equation for each of the velocity component $\tilde{\boldsymbol{u}}^{k+1}$ in (7.4.2) and a Poisson equation for ϕ^{k+1} in (7.4.7).

Remark 7.4.1 If part of the boundary is open, i.e., the problem is prescibed with the following boundary conditions:

$$\boldsymbol{u}|_{\Gamma_1}=\boldsymbol{h}_1,\ \boldsymbol{n}^t(\nu\nabla\boldsymbol{u}-pI)|_{\Gamma_2}=\boldsymbol{h}_2,\ \ \partial\Omega=\Gamma_1\cup\Gamma_2, \tag{7.4.8}$$

the above scheme should be modified as follows [69]:

$$\begin{cases}\dfrac{1}{2\delta t}(3\tilde{\boldsymbol{u}}^{k+1}-4\boldsymbol{u}^k+\boldsymbol{u}^{k-1})-\nu\Delta\tilde{\boldsymbol{u}}^{k+1}+\nabla p^k=\boldsymbol{g}(t_{k+1}),\\ \tilde{\boldsymbol{u}}^{k+1}|_{\Gamma_1}=\boldsymbol{h}_1^{k+1},\ \boldsymbol{n}^t(\nu\nabla\tilde{\boldsymbol{u}}^{k+1}-p^kI)|_{\Gamma_2}=\boldsymbol{h}_2^{k+1},\end{cases} \tag{7.4.9}$$

$$\begin{cases}\dfrac{1}{2\delta t}(3\boldsymbol{u}^{k+1}-3\tilde{\boldsymbol{u}}^{k+1})+\nabla\phi^{k+1}=0;\quad \nabla\cdot\boldsymbol{u}^{k+1}=0,\\ \boldsymbol{u}^{k+1}\cdot n|_{\Gamma_1}=\boldsymbol{h}_1^{k+1}\cdot n,\ \phi^{k+1}|_{\Gamma_2}=0;\end{cases} \tag{7.4.10}$$

and

$$p^{k+1}=\phi^{k+1}+p^k-\nu\nabla\cdot\tilde{\boldsymbol{u}}^{k+1}. \tag{7.4.11}$$

A second-order consistent splitting scheme

Although the rotational pressure-correction scheme is quite accurate, it still suf-

fers from a splitting error of order $\delta t^{\frac{3}{2}}$ for the H^1-norm of the velocity and L^2-norm of the pressure. We present below a consistent splitting scheme which removes this splitting error. The key idea behind the consistent splitting schemes is to evaluate the pressure by testing the momentum equation against gradients. By taking the L^2-inner product of the momentum equation in (7.4.1) with ∇q and noticing that $(\boldsymbol{u}_t, \nabla q) = -(\nabla \cdot \boldsymbol{u}_t, q)$, we obtain

$$\int_\Omega \nabla p \cdot \nabla q = \int_\Omega (\boldsymbol{f} + \nu \Delta \boldsymbol{u} - \boldsymbol{u} \cdot \nabla \boldsymbol{u}) \cdot \nabla q, \quad \forall q \in H^1(\Omega). \tag{7.4.12}$$

We note that if $\boldsymbol{u}$ is known, (7.4.12) is simply the weak form of a Poisson equation for the pressure. So, the principle we shall follow is to compute the velocity and the pressure in two consecutive steps: First, we evaluate the velocity by making explicit the pressure, then we evaluate the pressure by making use of (7.4.12).

Denoting $\boldsymbol{g}^{k+1} = \boldsymbol{f}^{k+1} - \big(2\boldsymbol{u}^n \cdot \nabla \boldsymbol{u}^n - \boldsymbol{u}^{n-1} \cdot \nabla \boldsymbol{u}^{n-1}\big)$, a formally second-order semi-implicit splitting scheme can be constructed as follows: find $\boldsymbol{u}^{k+1}$ and p^{k+1} such that

$$\frac{3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}}{2\Delta t} - \nu \Delta \boldsymbol{u}^{k+1} + \nabla(2p^k - p^{k-1}) = \boldsymbol{g}^{k+1}, \quad \boldsymbol{u}^{k+1}|_{\partial\Omega} = 0, \tag{7.4.13}$$

$$(\nabla p^{k+1}, \nabla q) = (\boldsymbol{g}^{k+1} + \nu \Delta \boldsymbol{u}^{k+1}, \nabla q), \quad \forall q \in H^1(\Omega). \tag{7.4.14}$$

Notice that we can use (7.4.13) to replace $\boldsymbol{g}^{k+1} + \nu \Delta \boldsymbol{u}^{k+1}$ in (7.4.14) by $(3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1})/(2\Delta t) + \nabla(2p^k - p^{k-1})$, leading to an equivalent formulation of (7.4.14):

$$(\nabla(p^{k+1} - 2p^k + p^{k-1}), \nabla q) = (\frac{3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}}{2\Delta t}, \nabla q), \quad \forall q \in H^1(\Omega). \tag{7.4.15}$$

We observe that if the domain Ω is sufficiently smooth, the solution of the above problem satisfies the following Poisson equation:

$$\begin{aligned} &-\Delta(p^{k+1} - 2p^k + p^{k-1}) = -\nabla \cdot (\frac{3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}}{2\Delta t}); \\ &\frac{\partial}{\partial \boldsymbol{n}}(p^{k+1} - 2p^k + p^{k-1})|_{\partial\Omega} = 0. \end{aligned} \tag{7.4.16}$$

Since the exact pressure does not satisfy any prescibed boundary condition, it is clear that the pressure approximation from (7.4.16) is plaqued by the artificial Neumann boundary condition which limits its accuracy. However, this defect can be easily overcome by using the identity $\Delta \boldsymbol{u}^{k+1} = \nabla\nabla \cdot \boldsymbol{u}^{k+1} - \nabla \times \nabla \times \boldsymbol{u}^{k+1}$, and replacing

$\Delta \boldsymbol{u}^{k+1}$ in (7.4.14) by $-\nabla\times\nabla\times\boldsymbol{u}^{k+1}$. this procedure amounts to removing in (7.4.14) the term $\nabla\nabla\cdot\boldsymbol{u}^{k+1}$. It is clear that this is a consistent procedure since the exact velocity is divergence-free. Thus, (7.4.14) should be replaced by

$$(\nabla p^{k+1}, \nabla q) = (\boldsymbol{g}^{k+1} - \nu\nabla\times\nabla\times\boldsymbol{u}^{k+1}, \nabla q),\ \forall q \in H^1(\Omega). \tag{7.4.17}$$

Once again, we can use (7.4.13) to reformulate (7.4.17) by replacing $\boldsymbol{g}^{k+1}-\nu\nabla\times\nabla\times\boldsymbol{u}^{k+1}$ with $(3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1})/2\Delta t + \nabla(2p^k - p^{k-1}) - \nu\nabla\nabla\cdot\boldsymbol{u}^{k+1}$. Thus, the second-order consistent splitting scheme takes the form

$$\frac{3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}}{2\Delta t} - \nu\Delta\boldsymbol{u}^{k+1} + \nabla(2p^k - p^{k-1}) = \boldsymbol{g}^{k+1},\ \ \boldsymbol{u}^{k+1}|_{\partial\Omega} = 0,$$

$$(\nabla\psi^{k+1}, \nabla q) = (\frac{3\boldsymbol{u}^{k+1} - 4\boldsymbol{u}^k + \boldsymbol{u}^{k-1}}{2\Delta t}, \nabla q), \quad \forall q \in H^1(\Omega), \tag{7.4.18}$$

with

$$p^{k+1} = \psi^{k+1} + (2p^k - p^{k-1}) - \nu\nabla\cdot\boldsymbol{u}^{k+1}. \tag{7.4.19}$$

Ample numerical results presented in [68] indicate that this scheme provides truly second-order accurate approximation for both the velocity and the pressure. However, a rigorous proof of this statement is still not available (cf. [69]).

Full discretization

It is straightforward to discretize in space the two schemes presented above. For a rectangular domain, we can use, for instance, the spectral-Galerkin method described in Chapter 6. To fix the idea, let $\Omega = (-1, 1)^d$ and set

$$X_N = P_N^d \cap H_0^1(\Omega)^d, \quad M_N = \{q \in P_{N-2} : \int_\Omega q = 0\}. \tag{7.4.20}$$

Then, the scheme (7.4.2)–(7.4.5) can be implemented as follows:

- **Step 1** Find $\tilde{\boldsymbol{u}}_N^{k+1} \in X_N$ such that

$$\begin{aligned}
&\frac{3}{2\delta t}(\tilde{\boldsymbol{u}}_N^{k+1}, \boldsymbol{v}_N) + \nu(\nabla\tilde{\boldsymbol{u}}_N^{k+1}, \nabla\boldsymbol{v}_N)\\
=&\frac{1}{2\delta t}(4\boldsymbol{u}_N^k - \boldsymbol{u}_N^{k-1} - \nabla(2p_N^k - p_N^{k-1}), \boldsymbol{v}_N)\\
&+ (I_N(\boldsymbol{f}^{k+1} - 2\boldsymbol{u}_N^k\cdot\nabla\boldsymbol{u}_N^k + \boldsymbol{u}_N^{k-1}\cdot\nabla\boldsymbol{u}_N^{k-1}), \boldsymbol{v}_N), \quad \forall\boldsymbol{v}_N \in X_N;
\end{aligned} \tag{7.4.21}$$

- **Step 2** Find $\phi_N^{k+1} \in M_N$ such that

$$(\nabla\phi_N^{k+1}, \nabla q_N) = \frac{3}{2\delta t}(\tilde{\boldsymbol{u}}_N^{k+1}, \nabla q_N), \quad \forall q_N \in M_N; \tag{7.4.22}$$

- **Step 3** Set

$$\begin{aligned} \boldsymbol{u}_N^{k+1} &= \tilde{\boldsymbol{u}}_N^{k+1} - \frac{2\delta t}{3}\nabla\phi_N^{k+1}, \\ p_N^{k+1} &= \phi_N^{k+1} + p_N^k - \nu\nabla\cdot\tilde{\boldsymbol{u}}_N^{k+1}. \end{aligned} \tag{7.4.23}$$

The scheme (7.4.18) and (7.4.19) can be implemented in a similar way:

- **Step 1** Find $\boldsymbol{u}_N^{k+1} \in X_N$ such that

$$\begin{aligned} &\frac{3}{2\delta t}(\boldsymbol{u}_N^{k+1}, \boldsymbol{v}_N) + \nu(\nabla\boldsymbol{u}_N^{k+1}, \nabla\boldsymbol{v}_N) \\ &= \frac{1}{2\delta t}(4\boldsymbol{u}_N^k - \boldsymbol{u}_N^{k-1} - (2\nabla p_N^k - p_N^{k-1}), \boldsymbol{v}_N) \\ &\quad + (I_N(\boldsymbol{f}^{k+1} - 2\boldsymbol{u}_N^k\cdot\nabla\boldsymbol{u}_N^k + \boldsymbol{u}_N^{k-1}\cdot\nabla\boldsymbol{u}_N^{k-1}), \boldsymbol{v}_N), \quad \forall\boldsymbol{v}_N \in X_N; \end{aligned} \tag{7.4.24}$$

- **Step 2** Find $\phi_N^{k+1} \in M_N$ such that

$$(\nabla\phi_N^{k+1}, \nabla q_N) = \frac{1}{2\delta t}(3\boldsymbol{u}_N^{k+1} - 4\boldsymbol{u}_N^k + \boldsymbol{u}_N^{k-1}, \nabla q_N), \qquad \forall q_N \in M_N; \tag{7.4.25}$$

- **Step 3** Set

$$p_N^{k+1} = \phi_N^{k+1} + 2p_N^k - p_N^{k-1} - \nu\Pi_{N-2}\nabla\cdot\boldsymbol{u}_N^{k+1}. \tag{7.4.26}$$

Hence, at each time step, the two spectral-projection schemes presented above only involve a vector Poisson equation for the velocity and a scalar Poisson equation for the pressure.

In this section, we only discussed the spectral-projection method for the Navier-Stokes equations. For numerical solutions of the Navier-Stokes equations; relevant papers using spectral Galerkin/finite element methods include [130], [163], [132], [116], [147], [44], while spectral collocation methods are treated in [95], [38], [87], [88], [86].

Exercise 7.4

Problem 1 Write a program implementing the rotational pressure-correction scheme and consistent splitting scheme using P_N for the velocity and P_{N-2} for the pressure.

Consider the exact solution of (7.4.1) $(\boldsymbol{u}, p)$ to be

$$\boldsymbol{u}(x, y, t) = \pi \sin t(\sin 2\pi y \sin^2 \pi x, -\sin 2\pi x \sin^2 \pi y),$$
$$p(x, y, t) = \sin t \cos \pi x \sin \pi y.$$

Compare the errors of the velocity and pressure at time $t = 1$ in both the L^2-norm and H^1-norm using the two schemes with $N = 32$ for $\delta t = 0.1, 0.05, 0.025, 0.0125$. Explain your results.

Problem 2 Use the rotatioanal pressure correction scheme to compute the steady state solution of the regularized driven cavity problem, i.e., $\Omega = (0, 1)^2$ with the boundary condition

$$\boldsymbol{u}|_{y=1} = (16x^2(1 - x^2), 0), \quad \boldsymbol{u}|_{\partial\Omega\setminus\{y=1\}} = 0.$$

Take $N = 32$ and $Re = 1/\nu = 400$. Compare your results with th benchmark results in [138].

7.5 Axisymmetric flows in a cylinder

Governing equations and time discretization
Spatial discretization
Treatment of the singular boundary condition
Numerical results

In this section, we apply the spectral-projection method presented in the last section to simulate an incompressible flow inside a cylinder. We assume that the flow is axisymmetric so we are effectively dealing with a two-dimensional problem. For more detail on the physical background of this problem and its numerical simulations, we refer to [106], [25], [109], [111], [108].

Governing equations and the time discretization

Consider a flow in an enclosed cylinder with the height H and radius R. The flow is driven by a bottom rotation rate of Ω rad s^{-1}. We shall non-dimensionalize the governing equations with the radius of the cylinder R as the length scale and $1/\Omega$ as the time scale. The Reynolds number is then $Re = \Omega R^2/\nu$, where ν is the kinematic viscosity. The flow is governed by another non-dimensional parameter, the aspect ratio of the cylinder $\Lambda = H/R$. Therefore, the domain for the space variables (r, z) is the rectangle

$$\mathcal{D} = \{(r, z) : r \in (0, 1) \text{ and } z \in (0, \Lambda)\}.$$

Let (u, v, w) be the velocity field in the cylindrical polar coordinates (r, θ, z) and assume the flow is axisymmetric, i.e., independent of the azimuthal θ direction, the Navier-Stokes equations (7.4.1) governing this axisymmetric flow in the cylindrical polar coordinates reads (cf. [111])

$$u_t + uu_r + wu_z - \frac{1}{r}v^2 = -p_r + \frac{1}{Re}\left(\tilde{\nabla}^2 u - \frac{1}{r^2}u\right), \tag{7.5.1}$$

$$v_t + uv_r + wv_z + \frac{1}{r}uv = \frac{1}{Re}\left(\tilde{\nabla}^2 v - \frac{1}{r^2}v\right), \tag{7.5.2}$$

$$w_t + uw_r + ww_z = -p_z + \frac{1}{Re}\tilde{\nabla}^2 w, \tag{7.5.3}$$

$$\frac{1}{r}(ru)_r + w_z = 0, \tag{7.5.4}$$

where

$$\tilde{\nabla}^2 = \partial_r^2 + \frac{1}{r}\partial_r + \partial_z^2 \tag{7.5.5}$$

is the Laplace operator in axisymmetric cylindrical coordinates. The boundary conditions for the velocity components are zero everywhere except that (i) $v = r$ at $\{z = 0\}$ which is the bottom of the cylinder; and (ii) $u_r = 0$ at $\partial\mathcal{D}\backslash\{z = 0\}$.

To simplify the presentation, we introduce the following notations:

$$\tilde{\Delta} = \begin{pmatrix} \tilde{\nabla}^2 - 1/r^2, & 0, & 0 \\ 0, & \tilde{\nabla}^2 - 1/r^2, & 0 \\ 0, & 0, & \tilde{\nabla}^2 \end{pmatrix}, \quad \tilde{\nabla} = \begin{pmatrix} \partial_r \\ 0 \\ \partial_z \end{pmatrix},$$

$$\Gamma_1 = \{(r, z) : r \in (0, 1) \text{ and } z = 0\}, \quad \Gamma_2 = \{(r, z) : r = 0 \text{ and } z \in (0, \Lambda)\},$$

and rewrite the equations (7.5.1)–(7.5.4) in vector form,

$$\begin{aligned} &\mathbf{u}_t + \mathbf{N}(\mathbf{u}) = -\tilde{\nabla}p + \frac{1}{Re}\tilde{\Delta}\mathbf{u}, \\ &\tilde{\nabla}\cdot\mathbf{u} := \frac{1}{r}(ru)_r + w_z = 0, \\ &\mathbf{u}|_{\partial\mathcal{D}\backslash(\Gamma_1\cup\Gamma_2)} = \mathbf{0}, \quad \mathbf{u}|_{\Gamma_1} = (0, r, 0)^{\mathrm{T}}, \quad (u, v, w_r)^{\mathrm{T}}|_{\Gamma_2} = \mathbf{0}, \end{aligned} \tag{7.5.6}$$

where $\mathbf{u} = (u, v, w)^{\mathrm{T}}$ and $\mathbf{N}(\mathbf{u})$ is the vector containing the nonlinear terms in (7.5.1)–(7.5.3).

To overcome the difficulties associated with the nonlinearity and the coupling of

velocity components and the pressure, we adapt the following semi-implicit second-order rotational pressure-correction scheme (cf. Section 7.4) for the system of equations (7.5.6):

$$\begin{cases} \dfrac{1}{2\Delta t}(3\tilde{\mathbf{u}}^{k+1} - 4\mathbf{u}^k + \mathbf{u}^{k-1}) - \dfrac{1}{Re}\tilde{\Delta}\tilde{\mathbf{u}}^{k+1} = -\tilde{\nabla}p^k - (2\mathbf{N}(\mathbf{u}^k) - \mathbf{N}(\mathbf{u}^{k-1})), \\ \tilde{\mathbf{u}}^{k+1}|_{\partial\mathcal{D}\setminus(\Gamma_1\cup\Gamma_2)} = \mathbf{0}, \quad \tilde{\mathbf{u}}^{k+1}|_{\Gamma_1} = (0, r, 0)^{\mathrm{T}}, \quad (\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{w}_r^{k+1})^{\mathrm{T}}|_{\Gamma_2} = \mathbf{0}. \end{cases} \tag{7.5.7}$$

$$\begin{cases} \dfrac{3}{2\Delta t}(\mathbf{u}^{k+1} - \tilde{\mathbf{u}}^{k+1}) + \tilde{\nabla}\phi^{k+1} = \mathbf{0}, \\ \tilde{\nabla}\cdot\mathbf{u}^{k+1} = 0, \\ (\mathbf{u}^{k+1} - \tilde{\mathbf{u}}^{k+1})\cdot\boldsymbol{n}|_{\partial\mathcal{D}} = 0, \end{cases} \tag{7.5.8}$$

and

$$p^{k+1} = p^k + \phi^{k+1} - \frac{1}{Re}\tilde{\nabla}\cdot\mathbf{u}^{k+1}, \tag{7.5.9}$$

where Δt is the time step, $\boldsymbol{n}$ is the outward normal at the boundary, and $\tilde{\mathbf{u}}^{k+1} = (\tilde{u}^{k+1}, \tilde{v}^{k+1}, \tilde{w}^{k+1})^{\mathrm{T}}$ and $\mathbf{u}^{k+1} = (u^{k+1}, v^{k+1}, w^{k+1})^{\mathrm{T}}$ are respectively the intermediate and final approximations of $\mathbf{u}$ at time $t = (k+1)\Delta t$.

It is easy to see that $\tilde{\mathbf{u}}^{k+1}$ can be determined from (7.5.7) by solving three Helmholtz-type equations. Instead of solving for $(\mathbf{u}^{k+1}, \phi^{k+1})$ from the coupled first-order differential equations (7.5.8), we apply the operator "$\tilde{\nabla}\cdot$" (see the definition in (7.5.6)) to the first equation in (7.5.8) to obtain an equivalent system

$$\begin{aligned} &\tilde{\nabla}^2\phi^{k+1} = \frac{3}{2\Delta t}\tilde{\nabla}\cdot\tilde{\mathbf{u}}^{k+1}, \\ &\partial_{\boldsymbol{n}}\phi^{k+1}|_{\partial\mathcal{D}} = 0, \end{aligned} \tag{7.5.10}$$

and

$$\mathbf{u}^{k+1} = \tilde{\mathbf{u}}^{k+1} - \frac{2\Delta t}{3}\tilde{\nabla}\phi^{k+1}. \tag{7.5.11}$$

Thus, $(\mathbf{u}^{k+1}, \phi^{k+1})$ can be obtained by solving an additional Poisson equation (7.5.10).

Next, we apply the spectral-Galerkin method for solving these equations.

Spatial discretization

We first transform the domain $\mathcal{D}$ to the unit square $\mathcal{D}^* = (-1,1)\times(-1,1)$ by using the transformations $r = (y+1)/2$ and $z = \Lambda(x+1)/2$. Then, at each time step, the systems (7.5.7) and (7.5.10) lead to the following four Helmholtz-type equations:

$$\begin{aligned} &\alpha u - \beta u_{xx} - \frac{1}{y+1}((y+1)u_y)_y + \frac{\gamma}{(y+1)^2}u = f \text{ in } \mathcal{D}^*, \\ &u|_{\partial\mathcal{D}^*} = 0; \end{aligned} \tag{7.5.12}$$

$$\begin{aligned} &\alpha v - \beta v_{xx} - \frac{1}{y+1}((y+1)v_y)_y + \frac{\gamma}{(y+1)^2}v = g \text{ in } \mathcal{D}^*, \\ &v|_{\partial\mathcal{D}^*\setminus\Gamma_1^*} = 0, \quad v|_{\Gamma_1^*} = \frac{1}{2}(y+1); \end{aligned} \tag{7.5.13}$$

$$\begin{aligned} &\alpha w - \beta w_{xx} - \frac{1}{y+1}((y+1)w_y)_y = h, \text{ in } \mathcal{D}^*, \\ &w|_{\partial\mathcal{D}^*\setminus\Gamma_2^*} = 0, \quad w_r|_{\Gamma_2^*} = 0; \end{aligned} \tag{7.5.14}$$

and

$$\begin{aligned} &-\beta p_{xx} - \frac{1}{y+1}((y+1)p_y)_y = q \text{ in } \mathcal{D}^*, \\ &\partial_{\boldsymbol{n}} p|_{\partial\mathcal{D}^*} = 0. \end{aligned} \tag{7.5.15}$$

In the above, $\Gamma_1^* = \{(x,y) : x = -1 \text{ and } y \in (-1,1)\}$, $\Gamma_2^* = \{(x,y) : x \in (-1,1) \text{ and } y = -1\}$, $\alpha = \frac{3}{8}Re/\Delta t$, $\beta = \Lambda^{-2}$, $\gamma = 1$, and $f,\ g,\ h,\ q$ are known functions depending on the solutions at the two previous time steps.

The spectral-Galerkin method of [142] can be directly applied to (7.5.12)–(7.5.15). We shall discuss the method for solving (7.5.12) in some detail. The other three equations can be treated similarly.

Let P_K be the space of all polynomials of degree less than or equal to K and set $P_{NM} = P_N \times P_M$. We set

$$X_{NM} = \{w \in P_{NM} : w|_{\partial\mathcal{D}^*} = 0\}.$$

Then the spectral-Galerkin method for (7.5.12) is to find $u_{_{NM}} \in X_{NM}$ such that

$$\begin{aligned} &\alpha\big((y+1)u_{_{NM}}, v\big)_{\tilde{\omega}} - \beta\big((y+1)\partial_x^2 u_{_{NM}}, v\big)_{\tilde{\omega}} - \Big(\big((y+1)\partial_y u_{_{NM}}\big)_y, v\Big)_{\tilde{\omega}} \\ &\quad + \gamma\big(\frac{1}{y+1}u_{_{NM}}, v\big)_{\tilde{\omega}} = \big((y+1)f, v\big)_{\tilde{\omega}}, \quad \forall\, v \in X_{NM}, \end{aligned} \tag{7.5.16}$$

where $(u,v)_{\tilde{\omega}} = \int_{\mathcal{D}^*} u\, v\, \omega(x)\, \omega(y)\, \mathrm{d}x\mathrm{d}y$ with $\omega(s)$ to be respectively 1 or $(1 - s^2)^{-\frac{1}{2}}$, depending on whether Legendre or Chebyshev polynomials are used. The

equation (7.5.16) is derived by first multiplying (7.5.12) by $(y+1)\omega(x)\omega(y)$ and then integrating over $\mathcal{D}^*$. The multiplication by $(y+1)$ is natural since the Jacobian of the transformation from the Cartesian coordinates to cylindrical coordinates is $r = ((y+1)/2)$ in the axisymmetric case. Since $u_{_{NM}} = 0$ at $y = -1$, we see that all terms in (7.5.16) are well defined and that no singularity is present.

For this problem, it is easy to verify that

$$X_{NM} = \text{span}\{\phi_i(x)\rho_j(y) : i = 0, 1, \cdots, N-2;\ j = 0, 1, \cdots, M-2\},$$

with $\phi_l(s) = \rho_l(s) = p_l(s) - p_{l+2}(s)$ where $p_l(s)$ is either the l-th degree Legendre or Chebyshev polynomial. Set

$$u_{_{NM}} = \sum_{i=0}^{N-2}\sum_{j=0}^{M-2} u_{ij}\phi_i(x)\rho_j(y),$$

and

$$\begin{aligned}
a_{ij} &= \int_{-1}^{1} \phi_j(x)\,\phi_i(x)\,\omega(x)\,\mathrm{d}x, \quad b_{ij} = -\int_{-1}^{1} \phi_j''(x)\,\phi_i(x)\,\omega(x)\,\mathrm{d}x,\\
c_{ij} &= \int_{-1}^{1} (y+1)\,\rho_j(y)\,\rho_i(y)\,\omega(y)\,\mathrm{d}y,\\
d_{ij} &= -\int_{-1}^{1} ((y+1)\,\rho_j'(y))'\,\rho_i(y)\,\omega(y)\,\mathrm{d}y,\\
e_{ij} &= \int_{-1}^{1} \frac{1}{y+1}\,\rho_j(y)\,\rho_i(y)\,\omega(y)\,\mathrm{d}y,\\
f_{ij} &= \int_{\mathcal{D}^*} (y+1) f\,\rho_j(y)\phi_i(x)\,\omega(x)\,\omega(y)\,\mathrm{d}x\mathrm{d}y,
\end{aligned} \tag{7.5.17}$$

and let $A,\ B,\ C,\ D,\ E,\ F$ and U be the corresponding matrices with entries given above. Then (7.5.16) is equivalent to the matrix system

$$\alpha AUC + \beta BUC + AUD + \gamma AUE = F. \tag{7.5.18}$$

Note that e_{ij} is well defined in spite of the term $\frac{1}{y+1}$, since $\rho_i(-1) = 0$. In the Legendre case, the matrices A, B, C, D, and E are all symmetric and sparsely banded.

Treatment of the singular boundary condition

The boundary condition for v is discontinuous at the lower right corner ($r =$

$1, z = 0)$. This singular boundary condition is a mathematical idealization of the physical situation, where there is a thin gap over which v adjusts from 1.0 on the edge of the rotating endwall to 0.0 on the sidewall. Therefore, it is appropriate to use a regularized boundary condition (so that v is continuous) which is representative of the actual gap between the rotating endwall and the stationary sidewall in experiments.

In finite difference or finite element schemes, the singularity is usually regularized over a few grid spacings in the neighborhood of the corner in an *ad hoc* manner. However, this simple treatment leads to a mesh-dependent boundary condition which in turn results in mesh-dependent solutions which prevents a sensible comparison between solutions with different meshes. Essentially, the grid spacing represents the physical gap size.

The singular boundary condition at $r = 1$ is

$$v(z) = 1 \text{ at } z = 0, \quad v(z) = 0 \quad \text{for } 0 < z \leqslant \Lambda,$$

which is similar to that of the driven cavity problem. Unless this singularity is treated appropriately, spectral methods may have severe difficulty dealing with it. In the past, most computations with spectral methods avoided this difficulty by using regularized boundary conditions which, unfortunately, do not approximate the physical boundary condition (e.g., [138], [38]). A sensible approach is to use the boundary layer function

$$v_\varepsilon(z) = \exp\left(-\frac{2z}{\Lambda\varepsilon}\right),$$

which has the ability to approximate the singular boundary condition to within any prescribed accuracy. Outside a boundary layer of width $\mathcal{O}(\varepsilon)$, $v_\varepsilon(z)$ converges to $v(z)$ exponentially as $\varepsilon \to 0$. However, for a given ε, approximately $\varepsilon^{-\frac{1}{2}}$ collocation points are needed to represent the boundary layer function v_ε. In other words, for a fixed number of modes M, we can only use $\varepsilon \geqslant \varepsilon(M)$ where $\varepsilon(M)$ can be approximately determined by comparing $I_M v_\varepsilon$ and v_ε, where $I_M v_\varepsilon$ is the polynomial interpolant of v_ε at the Gauss-Lobatto points.

Although it is virtually impossible to match the exact physical condition in the experimental gap region, the function v_ε with $\varepsilon = 0.006$ does provide a reasonable representation of the experimental gap. The function v_ε can be resolved spectrally with $M \geqslant M_\varepsilon$ modes, where M_ε is such that $I_M v_\varepsilon$ for a given ε is non-oscillatory. Due to the nonlinear term v^2/r in (7.5.1), we also require that $I_M v_{\varepsilon/2}$ be non-oscillatory (since $(v_\varepsilon)^2 = v_{\varepsilon/2}$). Figure 7.7(a) shows $I_M v_{0.006}$ for various M. It is clear that $I_{48} v_{0.006}$ is non-oscillatory. However, from Figure 7.7(b) we see that $I_{48} v_{0.003}$ is oscillatory near $z = 0$, while $I_{64} v_{0.003}$ is not. Thus, $M \approx 64$ is required for $\varepsilon = 0.006$.

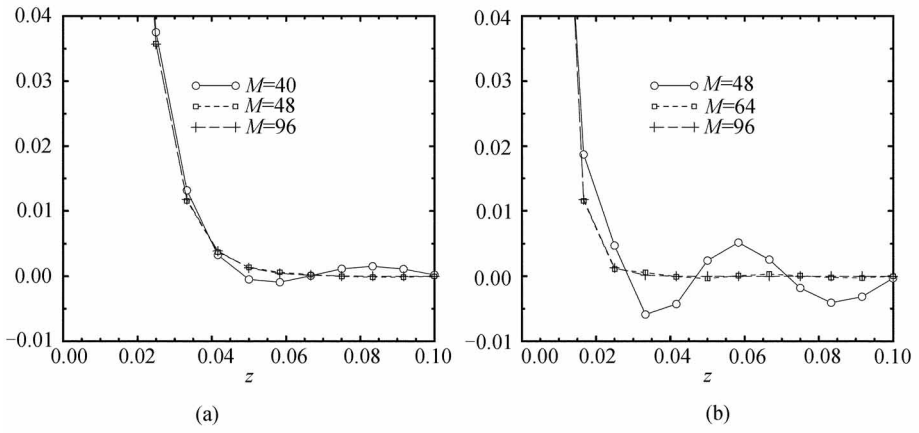

Figure 7.7

Variation of $I_M v_\varepsilon$ (with $\Lambda = 2.5$) in the vicinity of the singularity at $z = 0$ for (a) $\varepsilon = 0.006$ and (b) $\varepsilon = 0.003$, and various M as indicated.

Numerical results

For better visualization of the flow pattern, it is convenient to introduce the azimuthal vorticity η, the Stokes stream function ψ and the angular momentum Γ. These can be obtained from the velocity field (u, v, w) as follows:

$$\Gamma = rv; \quad \eta = u_z - w_r; \quad -\left(\partial_r^2 - \frac{1}{r}\partial_r + \partial_z^2\right)\psi = r\eta, \ \psi|_{\partial\mathcal{D}} = 0. \qquad (7.5.19)$$

Figure 7.8 shows plots of the solution for Stokes flow ($Re = 0$) for this problem. The governing equations (7.5.1)–(7.5.4) in the case $Re = 0$ reduce to

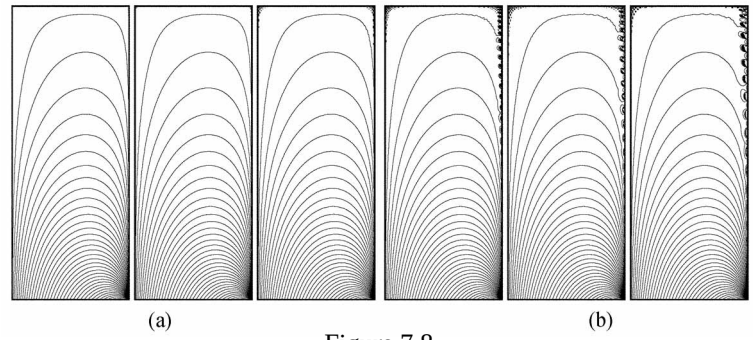

Figure 7.8

Contours of Γ for Stokes flow ($Re = 0$), using $v_{0.006}$ (a) and the *ad hoc* (b) regularization of the corner singularity. The leftmost plot in each set has $N = 56$, $M = 80$, the middle plots have $N = 48$, $M = 64$, and the right plots have $N = 40$, $M = 48$. All have been projected on to 201 uniform radial locations and 501 uniform axial locations.

Table 7.4

N, M	$\min(\Gamma)$ with $\varepsilon = 0.006$	$\min(\Gamma)$ with *ad hoc* B.C.
$56, 80$	-2.472×10^{-6}	-4.786×10^{-3}
$48, 64$	-9.002×10^{-6}	-6.510×10^{-3}
$40, 48$	-1.633×10^{-4}	-6.444×10^{-3}

Largest negative values of Γ on the grid points of a 201×501 uniform mesh, corresponding to the solutions for Stokes flow shown in Figure 7.8.

$$\tilde{\nabla}^2 v - \frac{1}{r^2} v = \tilde{\nabla}^2 \Gamma = 0,$$

with $\Gamma = 0$ on the axis, top endwall and sidewall, and $\Gamma = r^2$ on the rotating bottom endwall. The singular boundary condition on the sidewall has been regularized in Figure 7.8(a) with $v_{0.006}$ and in Figure 7.8(b) with the *ad hoc* method. For the solution of the Stokes problem with $\varepsilon = 0.006$, we judge that the error is acceptably small at $M = 64$ and is very small at $M = 80$. The measure of error used here is the largest value of negative Γ of the computed solution at the grid points of a uniform 201×501 mesh; the true solution has $\Gamma \geqslant 0$. These values are listed in Table 7.4. In contrast, with the *ad hoc* method the error does not decrease as M increases and the computed solutions exhibit large errors for all values of M considered.

We now present some numerical results using the spectral-projection scheme for $Re = 2494$ with $\Lambda = 2.5$. This Re is large enough that boundary layers are thin (thickness $\mathcal{O}(Re^{-\frac{1}{2}})$), but small enough that the flow becomes steady. The primary interests here are to determine the level of spatial resolution required for an asymptotically grid/mode independent solution, and to examine the accuracy of transients during the evolution to the steady state. We use rest as the initial condition and impulsively start the bottom endwall rotating at $t = 0$. This test case was well documented, both experimentally (cf. [43]) and numerically (cf. [107], [110]).

We begin by determining the level of resolution needed for a spectral computation of the case with $Re = 2494$, $\Lambda = 2.5$, and $\varepsilon = 0.006$. From the Stokes flow problem, we have seen that for $\varepsilon = 0.006$, the proper treatment of the singularity at the corner requires $M \approx 64$. Figure 7.9 shows the solutions at $t = 3000$, which are essentially at steady state (i.e. changes in any quantity being less than one part in 10^5 between successive time steps), from spectral computations using a variety of resolutions. The plots are produced by projecting the spectral solutions onto 201 radial and 501 axial uniformly distributed physical locations. A comparison of these contours shows very little difference, except for some oscillations in η, the azimuthal component of the vorticity, near the axis where $\eta \approx 0$. These oscillations are considerably reduced with an increase in the number of spectral modes used. Figure 7.10(a) presents a

detail time history of the azimuthal velocity at $(r, z) = (1/2, \Lambda/2)$, a point which is not particularly sensitive. It illustrates the convergence of the solutions as N and M are increased. It also demonstrates that the temporal characteristics of the flow transients are not sensitive to the level of spatial resolution.

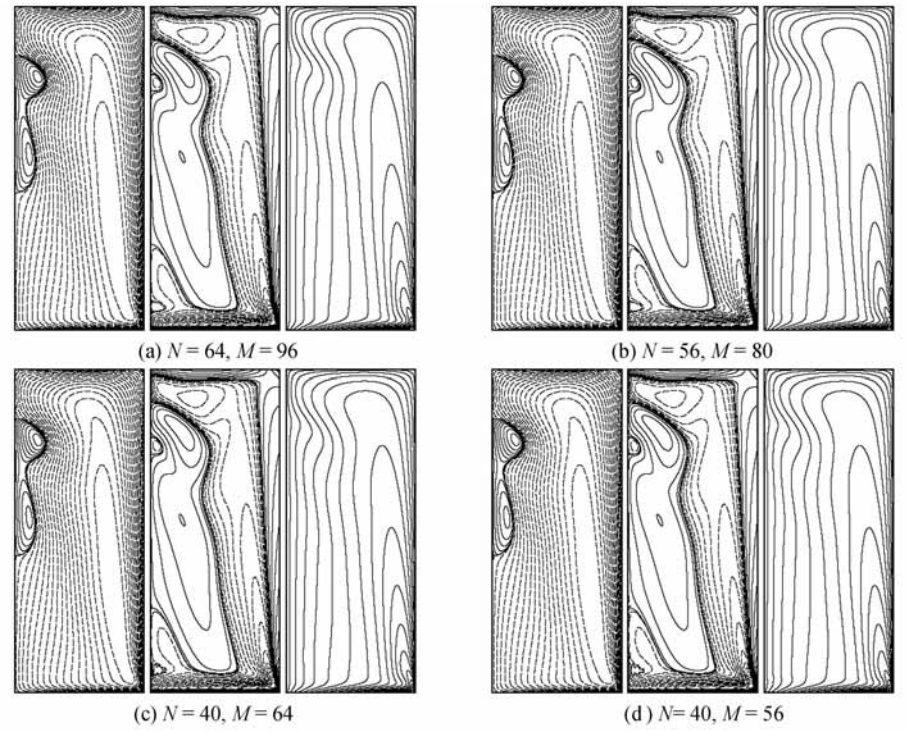

Figure 7.9

Contours of ψ, η, and Γ for $Re = 2494$ and $\Lambda = 2.5$ at $t = 3000$. Solutions are from spectral computations with $\Delta t = 0.04$ and $\varepsilon = 0.006$ and N and M as indicated. All have been projected on to 201 uniform radial locations and 501 uniform axial locations.

We have also computed cases with the same spatial resolution, but with two different temporal resolutions. Computations with $\Delta t = 0.04$ and $\Delta t = 0.01$ agree to four or five digits, which is of the same order as the time discretization error, and corresponding plots of the form shown in Figure 7.10(a) are indistinguishable for these cases.

In Figure 7.10(b), we show how the energy contribution E_k, from different levels of modes ($k = 0, \cdots, N$) decreases as k increases. E_k is defined as the sum of

the energy contribution from the modes v_{ik} for $i = 0, \cdots, M - N + k$ and $v_{k,j}$ for $j = 0, \cdots, k$ (v_{ij} are the coefficients of the Legendre expansion of v). The exponential decrease of E_k exhibited in Figure 7.10(b) is a good indication that the solutions are well resolved. Note also that except for a few of the highest modes, the energy distributions of differently resolved solutions overlap each other, providing another indication of their convergence.

From these convergence tests, we conclude that for $N = 40$, $M = 56$, $\Delta t = 0.04$, we already have very good results for the primitive variables (u, v, w) but the approximation for the azimuthal vorticity η at this resolution is not acceptable. We recall that η is computed by taking derivatives of u and w, so it is not unexpected that η requires more resolution than the velocity. At $N = 56$, $M = 80$, $\Delta t = 0.04$, the η contours are very smooth and this solution can be taken as being independent of discretization.

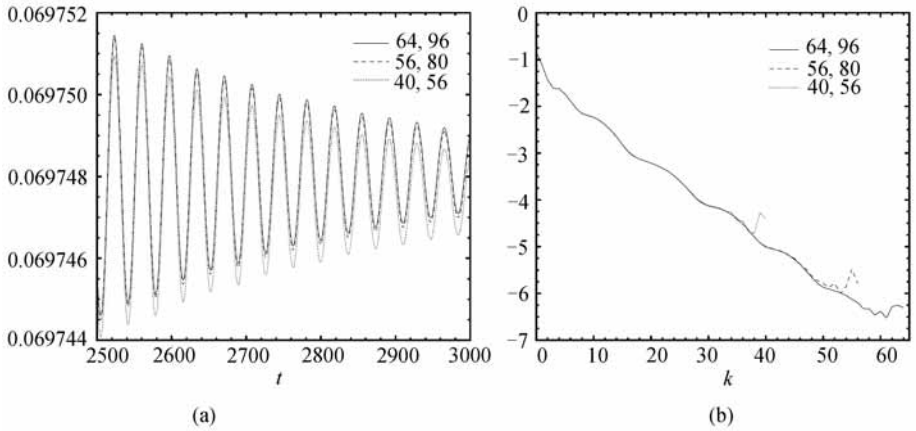

Figure 7.10

(a) Detail of the time history of $v(r = 1/2, z = \Lambda/2)$ for $Re = 2494$, $\Lambda = 2.5$, from spectral computations with $\varepsilon = 0.006$, and N and M as indicated. (b) $\log(E_k)$ versus k, where E_k is the energy contribution, from v, from different levels of modes $(k = 0, \cdots, M)$, corresponding to the solutions in left.

As a further illustration of the convergence of the solutions, we list in Table 7.5 the values and locations (on a 201×501 uniform physical grid for the spectral solutions, and on their own grids for the finite difference solutions) of three local maxima and minima of ψ and η.

For more details on these simulations, we refer to [111].

Table 7.5

N, M	ψ_1	ψ_2	ψ_3
	(r_1, z_1)	(r_2, z_2)	(r_3, z_3)
64, 96	7.6604×10^{-5}	-7.1496×10^{-3}	1.8562×10^{-5}
	(0.180, 1.96)	(0.760, 0.815)	(0.115, 1.36)
56, 80	7.6589×10^{-5}	-7.1495×10^{-3}	1.8578×10^{-5}
	(0.180, 1.96)	(0.760, 0.815)	(0.115, 1.36)
40, 56	7.6592×10^{-5}	-7.1498×10^{-3}	1.8582×10^{-5}
	(0.180, 1.96)	(0.760, 0.815)	(0.115, 1.36)
N, M	η_1	η_2	η_3
	(r_1, z_1)	(r_2, z_2)	(r_3, z_3)
64, 96	0.54488	-0.52342	-8.9785×10^{-3}
	(0.235, 2.04)	(0.335, 2.28)	(0.0500, 1.91)
56, 80	0.54488	-0.52343	-8.9797×10^{-3}
	(0.235, 2.04)	(0.335, 2.28)	(0.0500, 1.92)
40, 56	0.54502	-0.52341	-8.8570×10^{-3}
	(0.235, 2.04)	(0.335, 2.28)	(0.0500, 1.92)

Local maxima and minima of ψ and η, and their locations for $Re = 2494$, $\Lambda = 2.5$, and $\varepsilon = 0.006$, at $t = 3000$.

Appendix A

Some online software

Contents

A.1 MATLAB Differentiation Matrix Suite 300

A.2 PseudoPack 308

Differentiation matrices are derived from the spectral collocation (also known as pseudo-spectral) method for solving differential equations of boundary value type. This method is discussed in some detail in the last two chapters, but for more complete descriptions we refer to Canuto et al.[29], Fornberg[49], Fornberg and Sloan [50], Funaro[52], Gottlieb et al.[36], and Weideman and Reddy[168]. In the pseudo-spectral method the unknown solution to the differential equation is expanded as a global interpolant, such as a trigonometric or polynomial interpolant. In other methods, such as finite elements or finite differences, the underlying expansion involves local interpolants such as piecewise polynomials. In practice, this means that the accuracy of the spectral method is superior: for problems with smooth solutions convergence rates of $\mathcal{O}(e^{-cN})$ or $\mathcal{O}(e^{-c\sqrt{N}})$ are routinely achieved, where N is the number of degrees of freedom in the expansion (see, e.g., Canuto et al[29], Stenger [152]; Tadmor[156]). In contrast, finite elements or finite differences yield convergence rates that are only algebraic in N, typically $\mathcal{O}(N^{-2})$ or $\mathcal{O}(N^{-4})$.

There is, however, a price to be paid for using a spectral method instead of a finite element or a finite difference method: full matrices replace sparse matrices; stability restrictions may become more severe; and computer implementations, particularly for problems posed on irregular domains, may not be straightforward. Nevertheless,

provided the solution is smooth the rapid convergence of the spectral method often compensates for these shortcomings.

There are several general software packages for spectral computations, in FORTRAN or MATLAB. A FORTRAN package is written by Funaro[53] and available from

`http://cdm.unimo.it/home/matematica/funaro.daniele/rout.htm.`

This package provides subroutines for computing first and second derivative matrices and has support for general Jacobi polynomials, many quadrature formulas, and routines for computing expansion coefficients.

Another FORTRAN package, PseudoPack 2000 is written by Don and Costas[31] and available from

`http://www.cfm.brown.edu/people/wsdon/home.html.`

PseudoPack can compute up to fourth-order Fourier, Chebyshev, and Legendre collocation derivatives. Additional features include routines for filtering, coordinate mapping, and differentiation of functions of two and three variables.

Some MATLAB codes for spectral computations can be found in Trefethen[165]; the corresponding programs are available online at

`http://www.comlab.ox.ac.uk/oucl/work/nick.trefethen.`

where the readers will find many model problems in mechanics, vibrations, linear and nonlinear waves and other fields.

Another MATLAB package is the MATLAB Differentiation Matrix Suite written by Weideman and Reddy[168] and available from

`http://dip.sun.ac.za/~weideman/research/differ.html.`

In this appendix, we shall provide a rather detailed description for the MATLAB Differentiation Matrix Suite and PseudoPack 2000.

A.1 MATLAB Differentiation Matrix Suite

Below we present a rather detailed description to the MATLAB Differentiation Matrix Suite by Weideman and Reddy[168]. The suite consists of 17 MATLAB functions for solving differential equations by the spectral collocation (i.e., pseudo-spectral) method. It includes functions for computing derivatives of arbitrary order corresponding to Chebyshev, Hermite, Laguerre, Fourier, and sinc interpolants. Auxiliary functions are included for incorporating boundary conditions, performing interpola-

tion using barycentric formulas, and computing roots of orthogonal polynomials. It is demonstrated how to use the package for solving eigenvalue, boundary value, and initial value problems arising in the fields of special functions, quantum mechanics, nonlinear waves, and hydrodynamic stability.

The Differentiation Matrix Suite is available at

`http://ucs.orst.edu/~weidemaj/differ.html`

and at

`http://www.mathworks.com/support/ftp/diffeqv5.shtml`

in the Differential Equations category of the Mathworks user-contributed (MATLAB 5) M-file repository. The MATLAB functions in the suite are listed below:

a. **Differentiation Matrices (Polynomial Based)**
   (I) `poldif.m`: General differentiation matrices
   (II) `chebdif.m`: Chebyshev differentiation matrices
   (III) `herdif.m`: Hermite differentiation matrices
   (IV) `lagdif.m`: Laguerre differentiation matrices
b. **Differentiation Matrices (Nonpolynomial)**
   (I) `fourdif.m`: Fourier differentiation matrices
   (II) `sincdif.m`: Sinc differentiation matrices
c. **Boundary Conditions**
   (I) `cheb2bc.m`: Chebyshev second-derivative matrix incorporating Robin conditions
   (II) `cheb4c.m`: Chebyshev fourth-derivative matrix incorporating clamped conditions
d. **Interpolation**
   (I) `polint.m`: Barycentric polynomial interpolation at arbitrary distinct nodes
   (II) `chebint.m`: Barycentric polynomial interpolation at Chebyshev nodes
   (III) `fourint.m`: Barycentric trigonometric interpolation at equidistant nodes
e. **Transform-Based Derivatives**
   (I) `chebdifft.m`: FFT-based Chebyshev derivative
   (II) `fourdifft.m`: FFT-based Fourier derivative
   (III) `sincdifft.m`: FFT-based sinc derivative

f. **Roots of Orthogonal Polynomials**

(I) `legroots.m`: Roots of Legendre polynomials

(II) `lagroots.m`: Roots of Laguerre polynomials

(III) `herroots.m`: Roots of Hermite polynomials

g. **Examples**

(I) `cerfa.m`: Function file for computing the complementary error function with boundary condition (a) in (A.1)

(II) `cerfb.m`: Same as `cerfa.m`, but boundary condition (b) in (A.1) is used

(III) `matplot.m`: Script file for plotting the characteristic curves of Mathieu Rs equation

(IV) `ce0.m`: Function file for computing the Mathieu cosine-elliptic function

(V) `sineg.m`: Script file for solving the sine-Gordon equation

(VI) `sgrhs.m`: Function file for computing the right-hand side of the sine-Gordon system

(VII) `schrod.m`: Script file for computing the eigenvalues of the Schrödinger equation

(VIII) `orrsom.m`: Script file for computing the eigenvalues of the Orr -Sommerfeld equation

In the above, the boundary condition (A.1) is one of the two boundary conditions below:

$$y = 0 \text{ at } x = 1, \text{ or } y = 1 \text{ at } x = -1. \tag{A.1}$$

In Weideman and Reddy's software, they consider the case in which the set of interpolating functions $\{\phi_j(x)\}$ consists of polynomials of degree $N-1$. The two main functions in their suite, `poldif.m` and `chebdif.m`, deal with this situation. The former function computes differentiation matrices for arbitrary sets of points and weights; the latter function is restricted to Chebyshev nodes and constant weights.

The idea of a differentiation matrix of the spectral collocation method for solving differential equations is based on weighted interpolants of the form:

$$f(x) \approx p_{N-1}(x) = \sum_{j=1}^{N} \frac{\alpha(x)}{\alpha(x_j)} \phi_j(x) f_j. \tag{A.2}$$

Here $\{x_j\}_{j=1}^N$ is a set of distinct interpolation nodes; $\alpha(x)$ is a weight function; $f_j = f(x_j)$; and the set of interpolating functions $\{\phi_j(x)\}_{j=1}^N$ satisfies $\phi_j(x_k) = \delta_{jk}$ (the

Kronecker delta). This means that $p_{N-1}(x)$ defined by (A.2) is an interpolant of the function $f(x)$, in the sense that

$$f(x_k) = p_{N-1}(x_k), \quad k = 1, \cdots, N.$$

Associated with an interpolant such as (A.2) is the concept of a collocation derivative operator. This operator is generated by taking ℓ derivatives of (A.2) and evaluating the result at the nodes $\{x_k\}$:

$$f^{(\ell)}(x_k) \approx \sum_{j=1}^{N} \frac{d^\ell}{dx^\ell} \left[\frac{\alpha(x)}{\alpha(x_j)} \phi_j(x) \right]_{x=x_k} f_j, \quad k = 1, \cdots, N.$$

The derivative operator may be represented by a matrix $D^{(\ell)}$, the differentiation matrix, with entries

$$D^{(\ell)}_{k,j} = \frac{d^\ell}{dx^\ell} \left[\frac{\alpha(x)}{\alpha(x_j)} \phi_j(x) \right]_{x=x_k}. \tag{A.3}$$

The numerical differentiation process may therefore be performed as the matrix-vector product

$$\mathbf{f}^{(\ell)} = D^{(\ell)} \mathbf{f}, \tag{A.4}$$

where $\mathbf{f}$ (resp. $\mathbf{f}^{(\ell)}$) is the vector of function values (resp. approximate derivative values) at the nodes $\{x_k\}$.

The computation of spectral collocation differentiation matrices for derivatives of arbitrary order has been considered by Huang and Sloan[84], (constant weights) and Welfert [170](arbitrary $\alpha(x)$). The algorithm implemented in `poldif.m` and `chebdif.m` follows these references closely.

As discussed in Section 1.3 that in some cases (such as the set of Chebyshev points) explicit formulas are available, but this is the exception rather than the rule. The suite[168] has included three MATLAB functions for computing the zeros of the Legendre, Laguerre, and Hermite polynomials (called `legroots.m`, `lagroots.m`, and `herroots.m` respectively). The basis of these three functions is the three-term recurrence relation

$$\begin{aligned} &q_{n+1}(x) = (x - \alpha_n) q_n(x) - \beta_n^2 q_{n-1}(x), \quad n = 0, 1, \cdots, \\ &q_0(x) = 1, \quad q_{-1}(x) = 0. \end{aligned} \tag{A.5}$$

It is well known that the roots of the orthogonal polynomial $q_N(x)$ are given by the eigenvalues of the $N \times N$ tridiagonal Jacobi matrix

$$J = \begin{pmatrix} \alpha_0 & \beta_1 & & \\ \beta_1 & \alpha_1 & \beta_2 & \\ & & \ddots & \beta_{N-1} \\ & & \beta_{N-1} & \alpha_{N-1} \end{pmatrix}. \tag{A.6}$$

The coefficients (α_n, β_n) are given in the following table:

	Legendre	Laguerre	Hermite
α_n	0	$2n+1$	0
β_n	$n/\sqrt{4n^2-1}$	n^2	$1/2n$

Using MATLAB's convenient syntax the Jacobi matrix can easily be generated. For example, in the Legendre case this requires no more than three lines of code:

```
>>n = [1:N-1];
>>b = n./sqrt(4*n.^2-1);
>>J = diag(b,1) + diag(b,-1);
```

Once J has been created MATLAB's built-in `eig` routine can be used to compute its eigenvalues:

```
>>r = eig(J);
```

The functions `legroots.m`, `lagroots.m`, and `herroots.m` may be used in conjunction with `poldif.m` to generate the corresponding differentiation matrices. For example, in the Legendre case, assuming a constant weight the following two lines of code will generate first-and second-derivative matrices of order $N \times N$ on Legendre points:

```
>> x =  legroots(N);
>> D =  poldif(x,2);
```

Some calling commands for different basis functions are given below.

1. Chebyshev method

a. The calling command for `chebdif.m` is

```
>>[x, D] = chebdif(N, M);
```

On input the integer `N` is the size of the required differentiation matrices, and the integer `M` is the highest derivative needed. On output the vector `x`, of length N, contains the Chebyshev points

$$x_k = \cos\left((k-1)\pi/(N-1)\right), \qquad k = 1, \cdots, N, \tag{A.7}$$

and `D` is a $N \times N \times M$ array containing differentiation matrices $D^{(\ell)}$, $\ell = 1, \ldots, M$. It is assumed that $0 < M \leqslant N-1$.

b. The calling command for `chebint.m` is

```
>>p = chebint(f, x);
```

The input vector `f`, of length N, contains the values of the function $f(x)$ at the Chebyshev points (A.7). The vector `x`, of arbitrary length, contains the co-ordinates where the interpolant is to be evaluated. On output the vector `p` contains the corresponding values of the interpolant $p_{N-1}(x)$.

c. The calling command for `chebdifft.m` is

```
>>Dmf = chebdifft(f, M);
```

On input the vector `f`, of length N, contains the values of the function $f(x)$ at the Chebyshev points (A.7). M is the order of the required derivative. On output the vector `Dmf` contains the values of the Mth derivative of $f(x)$ at the corresponding points.

2. Hermite function

a. The calling command for `herdif.m` is

```
>>[x, D] = herdif(N, M, b);
```

On input the integer `N` is the size of the required differentiation matrices, and the integer `M` is the highest derivative needed. The scalar `b` is the scaling parameter b defined by the change of variable $x = b\tilde{x}$. On output the vector `x`, of length N, contains the Hermite points scaled by b. `D` is an $N \times N \times M$ array containing the differentiation matrices $D^{(\ell)}$, $\ell = 1, \cdots, M$.

b. The calling command for `herroots.m` is

```
>>r = herroots(N);
```

The input integer `N` is the degree of the Hermite polynomial, and the output vector `r` contains its N roots.

3. Laguerre function

a. The calling command for `lagdif.m` is

```
>>[x, D] = lagdif(N, M, b);
```

On input the integer `N` is the size of the required differentiation matrices, and the integer `M` is the highest derivative needed. The scalar `b` is the scaling parameter b discussed above. On output the vector `x`, of length N, contains the Laguerre points scaled by b, plus a node at $x = 0$. `D` is an $N \times N \times M$ array containing differentiation matrices $D^{(\ell)}$, $\ell = 1, \cdots, M$.

b. The calling command for `lagroots.m` is

```
>>r = lagroots(N);
```

The input integer `N` is the degree of the Laguerre polynomial, and the output vector `r` contains its N roots.

4. Fourier function

a. The calling command of `fourdif.m` is

```
>>[x, DM] = fourdif(N, M);
```

On input the integer `N` is the size of the required differentiation matrix, and the integer `M` is the derivative needed. On output, the vector `x`, of length N, contains the equispaced nodes given by (1.5.3), and `DM` is the $N \times N$ containing the differentiation matrix $D^{(M)}$. Unlike the other functions in the suite, `fourdif.m` computes only the single matrix $D^{(M)}$, not the sequence $D^{(1)}, \cdots, D^{(M)}$.

b. The calling command of `fourint.m` is

```
>>t = fourint(f, x)
```

On input the vector `f`, of length N, contains the function values at the equispaced nodes (1.5.3). The entries of the vector `x`, of arbitrary length, are the ordinates where

the interpolant is to be evaluated. On output the vector `t` contains the corresponding values of the interpolant $t_N(x)$ as computed by the formula (2.2.9a) or (2.2.9b).

c. The calling command for `fourdifft.m` is

```
>>Dmf = fourdifft(f, M);
```

On input the vector `f`, of length N, contains the values of the function $f(x)$ at the equispaced points (1.5.3); M is the order of the required derivative. On output the vector `Dmf` contains the values of the Mth derivative of $f(x)$ at the corresponding points.

The subroutine `cheb2bc.m` is a function to solve the general two-point boundary value problem

$$u''(x) + q(x)u'(x) + r(x)u(x) = f(x), \quad -1 < x < 1, \tag{A.8}$$

subject to the boundary conditions

$$a_+u(1) + b_+u'(1) = c_+, \quad a_-u(-1) + b_-u'(-1) = c_-. \tag{A.9}$$

It is assumed that a_+ and b_+ are not both 0, and likewise for a_- and b_-. The function `cheb2bc.m` generates a set of nodes $\{x_k\}$, which are essentially the Chebyshev points with perhaps one or both boundary points omitted. (When a Dirichlet condition is enforced at a boundary, that particular node is omitted, since the function value is explicitly known there.) The function also returns differentiation matrices $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$ which are the first- and second-derivative matrices with the boundary conditions (A.9) incorporated. The matrices $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$ may be computed from the Chebyshev differentiation matrices $D^{(1)}$ and $D^{(2)}$, which are computed by `chebdif.m`.

Table A.1 Solving the boundary value problem (A.10)

```
>>N = 16;
>>g = [2 -1 1; 2 1 -1];                    % Boundary condition array
>>[x, D2t, D1t, phip, phim] = cheb2bc(N, g); % Get nodes, matrices, and
                                           % vectors
>>f = 4*exp(x.^2);
>>p = phip(:,2)-2*x.*phip(:,1);            % psi+
>>m = phim(:,2)-2*x.*phim(:,1);            % psi-
>>D = D2t-diag(2*x)*D1t+2*eye(size(D1t));  % Discretization matrix
>>u = D\(f-p-m);                           % Solve system
```

The function `cheb2bc.m` computes the various matrices and boundary condition vectors described above. The calling command is

```
>>[x, D2t, D1t, phip, phim] = cheb2bc(N, g);
```

On input `N` is the number of collocation points used. The array `g = [ap bp cp; am bm cm]` contains the boundary condition coefficients, with a_+, b_+ and c_+ on the first row and a_-, b_- and c_- on the second. On output `x` is the node vector x. The matrices `D1t` and `D2t` contain $\tilde{D}^{(1)}$ and $\tilde{D}^{(2)}$, respectively. The first and second columns of `phip` contain $\tilde{\phi}'_+(x)$ and $\tilde{\phi}''_+(x)$, evaluated at the points in the node vector. Here $\tilde{\phi}'_\pm(x)$ are some modified basis functions, see [168]. Similarly, the first and second columns of `phim` contain $\tilde{\phi}'_-(x)$ and $\tilde{\phi}''_-(x)$, evaluated at points in the node vector. Since $\tilde{\phi}_+(x)$ and $\tilde{\phi}_-(x)$ are both 0 at points in the node vector, these function values are not returned by `cheb2bc.m`.

Using `cheb2bc.m`, it becomes a straightforward matter to solve the two-point boundary value problem (A.8) and (A.9). Consider, for example,

$$u'' - 2xu' + 2u = 4e^{x^2}, \quad 2u(1) - u'(1) = 1, \quad 2u(-1) + u'(-1) = -1. \tag{A.10}$$

The MATLAB code for solving (A.10) is given in Table A.1.

A.2 PseudoPack①

Global polynomial pseudo-spectral (or collocation) methods[29],[60] have been used extensively during the last decades for the numerical solution of partial differential equations (PDE). Some of the methods commonly used in the literatures are the Fourier collocation methods for periodical domain and the Jacobi polynomials with Chebyshev and Legendre polynomials as special cases for non-periodical domain. They have a wide range of applications ranging from 3-D seismic wave propagation, turbulence, combustion, non-linear optics, aero-acoustics and electromagnetics.

The underlying idea in those methods is to approximate the unknown solution in the entire computational domain by an interpolation polynomial at the quadrature (collocation) points. The polynomial is then required to satisfy the PDEs at the collocation points. This procedure yields a system of ODEs to be solved. These schemes can be very efficient as the rate of convergence (or the order of accuracy) depends only on the smoothness of the solution. This is known in the literature as *spectral*

① This section is kindly provided by Dr. W. S. Don of Brown University.

accuracy. In particular, if the solution of the PDE is analytic, the error decays exponentially. By contrast, in finite difference methods, the order of accuracy is fixed by the scheme.

While several software tools for the solution of partial differential equations (PDEs) exist in the commercial (e.g. DiffPack) as well as the public domain (e.g. PETSc), they are almost exclusively based on the use of low-order finite difference, finite element or finite volume methods. Geometric flexibility is one of their main advantages.

For most PDE solvers employing pseudo-spectral (collocation) methods, one major component of the computational kernel is the differentiation. The differentiation must be done accurately and efficiently on a given computational platform for a successful numerical simulation. It is not an easy task given the number of choice of algorithms for each new and existing computational platform.

Issues involving the complexity of the coding, efficient implementation, geometric restriction and lack of high quality software library tended to discourage the general use of the pseudo-spectral methods in scientific research and practical applications. In particularly, the lack of standard high quality library for pseudo-spectral methods forces individual researchers to build codes that were not optimal in terms of efficiency and accuracy.

Furthermore, while pseudo-spectral methods are at a fairly mature level, many critical issues regarding efficiency and accuracy have only recently been addressed and resolved. The knowledge of these solutions is not widely known and appears to restrict a more general usage of this class of algorithm in applications.

This package aims at providing to the user, in a high performance computing environment, a library of subroutines that provide an accurate, versatile, optimal and efficient implementation of the basic components of global pseudo-spectral methods on which to address a variety of applications of interest to scientists.

Since the user is shielded from any coding errors in the main computational kernels, reliability of the solution is enhanced. PseudoPack will speed up code development, increase scientific productivity and enhance code re-usability.

Major features of the PseudoPack library

PseudoPack is centered on subroutines for performing basic operations such as generation of proper collocation points, differentiation and filtering matrices. These routines provide a highly optimized computational kernel for pseudo-spectral methods based on either the Fourier series for periodical problems or the Chebyshev or

Legendre polynomials in simple non-periodical computational domain for the solution of initial-boundary value problems. State-of-the-art numerical techniques such as Even-Odd Decomposition[150] and specialized fast algorithms are employed to increase the efficiency of the library. Advance numerical algorithms, including accuracy enhancing mapping and filtering, are incorporated in the library.

The library contain a number of user callable routines that return the derivatives and/or filtering (smoothing) of, possibly multi-dimensional, data sets. As an application extension of the library, we have included routines for computing the conservative and non-conservative form of the derivative operators Gradient ∇, Divergence $\nabla\cdot$, Curl $\nabla\times$ and Laplacian ∇^2 operators in the 2D/3D general curvilinear coordination.

The source codes of the library is written in FORTRAN 90. The macro and conditional capability of C Preprocessor allows the software package be compiled into several versions with several different computational platforms. Several popular computational platforms (IBM RS6000, SGI Cray, SGI, SUN) are supported to take advantages of any existing optimized native library such as General Matrix-Matrix Multiply (GEMM) from Basic Linear Algebra Level 3 Subroutine (BLAS 3), Fast Fourier Transform (FFT) and Fast Cosine/Sine Transform (CFT/SFT). In term of flexibility and user interaction, any aspect of the library can be modified by minor change in a small number of input parameters.

Summary of major features

1. Derivatives of up to order four are supported for the Fourier, Chebyshev and Legendre collocation methods that are based on the Gauss-Lobatto, Gauss-Radau and Gauss quadrature nodes.

2. Matrix-Matrix Multiply, Even-Odd Decomposition and Fast Fourier Transform Algorithms are supported for computing the derivative/smoothing of a function.

3. Makefiles are available for compilation on system by IBM (RS/6000), SGI Cray, SGI, SUN and Generic UNIX machine.

4. Native fast assembly library calls such as General Matrix-Matrix Multiply (GEMM) from Basic Linear Algebra Level 3 Subroutine (BLAS 3), Fast Fourier Transform (FFT) and Fast Cosine/Sine Transform (CFT/SFT) when available, are deployed in the computational kernel of the PseudoPack.

5. Special fast algorithms, e.g. Fast Quarter-Wave Transform and Even-Odd Decomposition Algorithm, are provided for cases when the function has either even or odd symmetry.

6. Kosloff-Tal-Ezer mapping is used to reduce the round-off error for the Chebyshev and Legendre differentiation.

7. Extensive built-in and user-definable grid mapping function suitable for finite, semi-infinite and infinite domain are provided.

8. Built-in filtering (smoothing) of a function and its derivative are incorporated in the library.

9. Differentiation and smoothing can be applied to either the first or the second dimension of a two-dimensional data array.

10. Conservative and non-conservative form of Derivative operators, namely, Gradient ∇, Divergence $\nabla\cdot$, Curl $\nabla\times$ and Laplacian ∇^2 operators in the 2D/3D general curvilinear coordination using pseudo-spectral methods are available.

11. Memory usage by the PseudoPack is carefully minimized. User has some control over the amount of temporary array allocation.

12. Unified subroutine call interface allows modification of any aspect of the library with minor or no change to the subroutine call statement.

Illustration

As an illustration of the functionality of PseudoPack, we present a Pseudo-Code for computing the derivative of a two-dimensional data array f_{ij} using FORTRAN 90 language syntax.

The procedure essentially consists of four steps:

a. Specify all necessary non-default parameters and options that determine a specific spectral collocation scheme, for example, Chebyshev collocation method :

```
call PS_Setup_Property (Method=1)
```

b. Finds the storage requirement for the differentiation operator `D` :

```
call PS_Get_Operator_Size (M_D, ... )
ALLOCATE (D(M_D))
```

c. Setup the differentiation operator `D` :

```
call PS_Setup_Operator (D, ... )
```

d. Performs the differentiation operation by compute `D_f` = $\partial_x f$

```
call PS_Diff (D, f, D_f, ...)
```

These subroutine calls shall remain unchanged regardless of any changes made to the subroutine arguments such as `Method,Algorithm` etc. It provides to the user a uniform routine interface with which to work with. For example, to change the basis of approximation from Chebyshev polynomial to Legendre polynomial, it can be easily accomplished by changing the input parameter `Method=1` to `Method=2`.

Some general remarks

Some general remarks for the library are listed below:

1. The numerical solution of the PDE $U(x, y, t)$ at any given time t is stored in a two dimensional array `u(0:LDY-1,0:M)` with the leading dimension `LDY >= N`, where `N` and `M` are the number of collocation points in x and y direction respectively.

2. The suffix `_x` and `_y` denote the coordinate direction in which the variable is referring to.

3. The differentiation operator is `D` and the smoothing operator is `S` with the suffice `_x` and `_y` to denote the coordinate direction.

4. The name of subroutines with prefix `PS_` designates library routine calls to the PseudoPack library. Please consult the PseudoPack's manual for details.

5. The derived data type

`Property, Grid_Index, Domain, Mapping,Filtering_D, Filtering_S`

are used to store the specification of a specific differentiation operator.

6. The differentiation D and smoothing S operators are specified by calling the setup subroutine `PS_Setup`. The behavior of the operators can be modified by changing one or more optional arguments of the subroutine by changing the data in the respective derived data type such as `Property`.

To specify the Fourier, Chebyshev and Legendre method, one set `Method=0,1,2`, respectively.

To specify the matrix, Even-Odd Decomposition and Fast Transform Algorithm, one set `Algorithm=0,1,2`, respectively.

7. The important library calls are `PS_Diff` and `PS_Smooth` which perform the differentiation and smoothing according to the given differentiation and smoothing operators D and S as specified in `PS_Setup`.

A demonstration program for the use of PseudoPack can be found in
`http://www.cfm.brown.edu/people/wsdon/home.html`

Bibliography

[1] M. Abramowitz and I. A. Stegun. 1972. *Handbook of Mathematical Functions*. Dover, New York

[2] B. K. Alpert and V. Rokhlin. 1991. A fast algorithm for the evaluation of Legendre expansions. *SIAM J. Sci. Stat. Comput.*, 12:158–179

[3] B.-Y. Guo an L.-L. Wang. Jacobi approximations in certain Besov spaces. *To appear in J. Approx. Theory*

[4] K. Arrow, L. Hurwicz, and H. Uzawa. 1958. *Studies in Nonlinear Programming*. Stanford University Press

[5] U. Ascher, J. Christiansen, and R. D. Russell. 1981. Collocation software for boundary value odes. *ACM Trans. Math. Software*, 7:209–222

[6] K. E. Atkinson. 1997. *The Numerical Solution of Integral Equations of the Second Kind*. Cambridge Monographs on Applied and Computational Mathematics, Vol. 4. Cambridge University Press

[7] I. Babuska. 1972. The finite element method with Lagrangian multipliers. *Numer. Math.*, 20:179–192

[8] W. Bao, D. Jaksch, and P.A. Markowich. 2003. Numerical solution of the Gross-Pitaevskii equation for Bose-Einstein condensation. *J. Comput. Phys.*, 187

[9] W. Bao, S. Jin, and P.A. Markowich. 2003. Numerical study of time-splitting spectral discretizations of nonlinear Schrödinger equations in the semi-clasical regimes. *SIAM J. Sci. Comp.*, 25:27–64

[10] W. Bao and J. Shen. 2005. A fourth-order time-splitting Laguerre-Hermite pseudo-spectral method for Bose-Einstein condensates. *SIAM J. Sci. Comput.*, 26:2110–2028

[11] C. Bernardi and Y. Maday. 1997. Spectral method. In P. G. Ciarlet and L. L. Lions, editors, *Handbook of Numerical Analysis, V. 5 (Part 2)*. North-Holland

[12] C. Bernardi and Y. Maday. 1999. Uniform inf-sup conditions for the spectral discretization of the Stokes problem. *Math. Models Methods Appl. Sci.*, 9(3): 395–414

[13] G. Birkhoff and G. Fix. 1970. Accurate eigenvalue computation for elliptic problems. In *Numerical Solution of Field Problems in Continuum Physics, Vol 2, SIAM–AMS Proceedings*, pages 111–151. AMS, Providence

[14] P. Bjørstad. 1983. Fast numerical solution of the biharmonic Dirichlet problem on rectangles. *SIAM J. Numer. Anal.*, 20: 59–71

[15] P. E. Bjørstad and B. P. Tjostheim. 1997. Efficient algorithms for solving a fourth order equation with the spectral-Galerkin method. *SIAM J. Sci. Comput.*, 18

[16] J. L. Bona, S. M. Sun, and B. Y Zhang. A non-homogeneous boundary-value problem for the Korteweg-de Vries equation posed on a finite domain. *Submitted*.

[17] J. P. Boyd. 1982. The optimization of convergence for Chebyshev polynomial methods in an unbounded domain. *J. Comput. Phys.*, 45: 43–79

[18] J. P. Boyd. 1987. Orthogonal rational functions on a semi-infinite interval. *J. Comput. Phys.*, 70: 63–88

[19] J. P. Boyd. 1987. Spectral methods using rational basis functions on an infinite interval. *J. Comput. Phys.*, 69: 112–142

[20] J. P. Boyd. 1989. *Chebyshev and Fourier Spectral Methods, 1st ed.* Springer-Verlag, New York

[21] J. P. Boyd. 1992. Multipole expansions and pseudospectral cardinal functions: a new generation of the fast Fourier transform. *J. Comput. Phys.*, 103: 184–186

[22] J. P. Boyd. 2001. *Chebyshev and Fourier Spectral Methods, 2nd ed.* Dover, Mineola, New York

[23] F. Brezzi. 1974. On the existence, uniqueness and approximation of saddle-point problems arising from Lagrangian multipliers. *Rev. Française Automat. Informat. Recherche Opérationnelle Sér. Rouge*, 8(R-2): 129–151

[24] F. Brezzi and M. Fortin. 1991. *Mixed and Hybrid Finite Element Methods*. Springer-Verlag, New York

[25] G. L. Brown and J. M. Lopez. 1990. Axisymmetric vortex breakdown. II. Physical mechanisms. *J. Fluid Mech.*, 221: 553–576

[26] H. Brunner. 2004. *Collocation Methods for Volterra Integral and Related Functional Differential Equations*. Cambridge Monographs on Applied and Computational Mathematics, Vol. 15. Cambridge University Press

[27] J. C. Butcher. 1987. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. John Wiley & Sons, New York

[28] W. Cai, D. Gottlieb, and C.-W. Shu. 1989. Essentially nonoscillatory spectral Fourier methods for shock wave calculation. *Math. Comp.*, 52: 389–410

[29] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. A. Zang. 1988 *Spectral Methods for Fluid Dynamics*. Springer-Verlag, New York

[30] A. J. Chorin. 1968. Numerical solution of the Navier-Stokes equations. *Math. Comp.*, 22: 745–762

[31] B. Ciosta and W. S. Don. 1999. Pseudopack 2000: a spectral method libraray. See http://www.labma.ufrj.br/∼costa/PseudoPack2000/Main.htm.

[32] T. Colin and J.-M. Ghidaglia. 2001. An initial-boundary value problem for the Korteweg-de-Vries equation posed on a finite interval. *Adv. Diff. Eq.*, 6(12): 1463–1492

[33] J. W. Cooley and J. W. Tukey. 1965. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.*, 19: 297–301

[34] O. Coulaud, D. Funaro, and O. Kavian. 1990. Laguerre spectral approximation of elliptic problems in exterior domains. *Comp. Mech. in Appl. Mech and Eng.*, 80: 451–458

[35] M. Crandall and A. Majda. 1980. The method of fractional steps for conservation laws. *Numer. Math.*, 34: 285–314

[36] M. Y. Hussaini D. Gottlieb and S. A. Orszag. 1984. Theory and applications of spectral methods. In D. Gottlieb R. Voigt and M. Y. Hussaini, editors, *Spectral Methods for Partial Differential Equations.*, pages 1–54. SIAM

[37] P. J. Davis. 1975. *Interpolation and Approximation*. Dover Publications, New York

[38] P. Demaret and M. O. Deville. 1991. Chebyshev collocation solution of the Navier-Stokes equations using multi-domain decomposition and finite element preconditioning. *J. Comput. Phys.*, 95: 359–386

[39] S.C.R. Dennis and J. D. Hudson. 1989. Compact h^4 finite-difference approximations to operators of Navier-Stokes type. *J. Comput. Phys.*, 85: 390–416

[40] M. O. Deville, P. F. Fischer and E. H. Mund. 2002. *High-order methods for incompressible fluid flow*. Volume 9 of Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press

[41] W.S. Don and D. Gottlieb. 1994. The Chebyshev-Legendre method: implementing Legendre methods on Chebyshev points. *SIAM J. Numer. Anal.*, 31: 1519–1534

[42] H. Eisen, W. Heinrichs, and K. Witsch. 1991. Spectral collocation methods and polar coordinate singularities. *J. Comput. Phys.*, 96: 241–257

[43] M. P. Escudier. 1994. Observations of the flow produced in a cylindrical container by a rotating endwall. *Expts. Fluids*, 2: 189–196

[44] P. F. Fischer. 1997. An overlapping Schwarz method for spectral element solution of the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 133: 84–101

[45] J. C. M. Fok, B.-Y. Guo, and T. Tang. 2002. Combined Hermite spectral-finite difference method for the fokker-planck equations. *Math. Comp.*, 71: 1497–1528

[46] B. Fornberg. 1988. Generation of finite difference formulas on arbitrarily spaced grids. *Math. Comp.*, 51: 699–706

[47] B. Fornberg. 1990. An improved pseudospectral method for initial boundary value problems. *J. Comput. Phys.*, 91: 381–397

[48] B. Fornberg. 1995. A pseudospectral approach for polar and spherical geometries. *SIAM J. Sci. Comput.*, 16: 1071–1081

[49] B. Fornberg. 1996. *A Practical Guide to Pseudospectral Methods*. Cambridge University Press, New York

[50] B. Fornberg and D.M. Sloan. 1994. A review of pseudospectral methods for solving partial differential equations. In A. Iserles, editor, *Acta Numerica*, pages 203–267. Cambridge University Press, New York

[51] B. Fornberg and G. B. Whitham. 1978. A numerical and theoretical study of certain nonlinear wave phenomena. *Philosophical Transactions of the Royal Society of London*, 289: 373–404

[52] D. Funaro. 1992. *Polynomial Approxiamtions of Differential Equations*. Springer-verlag

[53] D. Funaro. Fortran routines for spectral methods, 1993. available via anonymous FTP at ftp.ian.pv.cnr.it in pub/splib.

[54] D. Funaro and O. Kavian. 1991. Approximation of some diffusion evolution equations in unbounded domains by Hermite functions. *Math. Comp.*, 57: 597–619

[55] V. Girault and P. A. Raviart. 1986. *Finite Element Methods for Navier-Stokes Equations*. Springer-Verlag

[56] R. Glowinski. 2003. Finite element methods for incompressible viscous flow. In *Handbook of numerical analysis, Vol. IX*, Handb. Numer. Anal., IX, pages 3–1176. North-Holland, Amsterdam

[57] S. K. Godunov. 1959. Finite difference methods for numerical computations of discontinuous solutions of the equations of fluid dynamics. *Mat. Sb.*, 47:271–295. in Russian

[58] D. Gottlieb and L. Lustman. 1983. The spectrum of the Chebyshev collocation operator for the heat equation. *SIAM J. Numer. Anal.*, 20: 909–921

[59] D. Gottlieb, L. Lustman, and S. A. Orszag. 1981. Spectral calculations of one-dimensional inviscid compressible flow. *SIAM J. Sci. Stat. Comput.*, 2: 286–310

[60] D. Gottlieb and S. A. Orszag. 1977. *Numerical Analysis of Spectral Methods*. SIAM, Philadelphia, PA

[61] D. Gottlieb and S. A. Orszag. 1977. *Numerical Analysis of Spectral Methods: Theory and Applications*. SIAM-CBMS, Philadelphia

[62] D. Gottlieb and C.-W. Shu.1994. Resolution properties of the Fourier method for discontinuous waves. *Comput. Methods Appl. Mech. Engrg.*, 116: 27–37

[63] D. Gottlieb and C.-W. Shu. 1997. On the gibbs phenomenon and its resolution. *SIAM Rev.*, 39(4): 644–668

[64] L. Greengard. 1991. Spectral integration and two-point boundary value problems. *SIAM J. Numer. Anal.*, 28:1071–1080

[65] L. Greengard and V. Rokhlin. 1987. A fast algorithm for particle simulations. *J. Comput. Phys.*, 73: 325–348

[66] C. E. Grosch and S. A. Orszag. 1977. Numerical solution of problems in unbounded regions: coordinates transforms. *J. Comput. Phys.*, 25: 273–296

[67] E.P. Gross. 1961. Structure of a quantized vortex in boson systems. *Nuovo. Cimento.*, 20: 454–477

[68] J. L. Guermond and J. Shen. 2003. A new class of truly consistent splitting schemes for incompressible flows. *J. Comput. Phys.*, 192(1): 262–276

[69] J.L. Guermond, P. Minev and J. Shen. An overview of projection methods for incompressible flows. *Inter. J. Numer. Methods Eng.*, to appear

[70] J.L. Guermond and J. Shen. On the error estimates of rotational pressure-correction projection methods. *To appear in Math. Comp*

[71] J.L. Guermond and J. Shen. 2004. On the error estimates of rotational pressure-correction projection methods. *Math. Comp*, 73: 1719–1737

[72] B.-Y. Guo. 1999. Error estimation of Hermite spectral method for nonlinear partial differential equations. *Math. Comp.*, 68: 1067–1078

[73] B.-Y. Guo, H.-P. Ma, and E. Tadmor. 2001. Spectral vanishing viscosity method for nonlinear conservation laws. *SIAM J. Numer. Anal.*, 39(4): 1254–1268

[74] B.-Y. Guo and J. Shen. 2000. Laguerre-Galerkin method for nonlinear partial differential equations on a semi-infinite interval. *Numer. Math.*, 86: 635–654

[75] B.-Y. Guo, J. Shen and L. Wang. Optimal spectral-Galerkin methods using generalized jacobi polynomials. *To appear in J. Sci. Comput*

[76] B.-Y. Guo, J. Shen, and Z. Wang. 2000. A rational approximation and its applications to differential equations on the half line. *J. Sci. Comp.*, 15: 117–147

[77] B.-Y. Guo and L.-L. Wang. 2001. Jacobi interpolation approximations and their applications to singular differential equations. *Adv. Comput. Math.*, 14: 227–276

[78] M. M. Gupta. 1991. High accuracy solutions of incompressible Navier-Stokes equations. *J. Comput. Phys.*, 93: 345–359

[79] D. B. Haidvogel and T. A. Zang. 1979. The accurate solution of Poisson's equation by expansion in Chebyshev polynomials. *J. Comput. Phys.*, 30: 167–180

[80] P. Haldenwang, G. Labrosse, S. Abboudi, and M. Deville. 1984. Chebyshev 3-d spectral and 2-d pseudospectral solvers for the helmholtz equation. *J. Comput. Phys.*, 55: 115–128

[81] P. Henrici. 1986. *Applied and Computational Complex Analysis*, volume 3. Wiley, New York

[82] M. Hestenes and E. Stiefel. 1952. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Stand.*, 49: 409–436

[83] M. H. Holmes. 1995. *Introduction to Perturbation Methods*. Springer-Verlag, New York

[84] W.-Z. Huang and D. Sloan. 1994. The pseudospectral method for solving differential eigenvalue problems. *J. Comput. Phys.*, 111: 399–409

[85] W.-Z. Huang and D. M. Sloan. 1993. Pole condition for singular problems: the pseudospectral approximation. *J. Comput. Phys.*, 107: 254–261

[86] W.-Z. Huang and T. Tang. 2000. Pseudospectral solutions for steady motion of a viscous fluid inside a circular boundary. *Appl. Numer. Math.*, 33: 167–173

[87] A. Karageorghis. 1992. The numerical solution of a laminar flow in a reentrant tube geometry by a Chebyshev spectral element collocation method. *Comput. Methods Appl. Mech. Engrg.*, 100: 339–358

[88] A. Karageorghis and T. N. Phillips. 1989. Spectral collocation methods for Stokes flow in contraction geometries and unbounded domains. *J. Comput. Phys.*, 80: 314–330

[89] G. E. Karniadakis and S. .J. Sherwin. 1999. *Spectral/hp Element Methods for CFD*. Oxford UniversityPress

[90] I. K. Khabibrakhmanov and D. Summers. 1998. The use of generalized Laguerre polynomials in spectral methods for nonlinear differential equations. *Comput. Math. Appl.*, 36: 65–70

[91] S. D. Kim and S. V. Parter. 1997. Preconditioning Chebyshev spectral collocation by finite difference operators. *SIAM J. Numer. Anal.*, 34, No. 3: 939–958

[92] D. Kincaid and E. W. Cheney. 1999. *Numerical Analysis, Mathematics of Scientific Computing*. Brooks/Cole, 3rd edition

[93] D. Kosloff and H. Tal-Ezer. 1993. A modified Chebyshev pseudospectral method with an $o(n^{-1})$ time step restriction. *J. Comput. Phys.*, 104: 457–469

[94] M. D. Kruskal and N. J. Zabusky. 1996. Exact invariants for a class of nonlinear wave equations. *J. Math. Phys.*, 7: 1256–1267

[95] H. C. Ku, R. S. Hirsh, and T. D. Taylor. 1987. A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations. *J. Comput. Phys.*, 70: 439–462

[96] P. D. Lax. 1978. Accuracy and resolution in the computation of solutions of linaer and nonlinear equations. In *Recent Advances in Numerical Analysis*, pages 107–117. Academic Press, London, New York

[97] P. D. Lax and B. Wendroff. 1960. Systems of conservation laws. *Commun. Pure Appl. Math.*, 13: 217–237

[98] P. Leboeuf and N. Pavloff. 2001. Bose-Einstein beams: Coherent propagation through a guide. *Phys. Rev. A*, 64: aritcle 033602

[99] J. Lee and B. Fornberg. 2003. A split step approach for the 3-d Maxwell's equations. *J. Comput. Appl. Math.*, 158: 485–505

[100] M. Lentini and V. Peyrera. 1977. An adaptive finite difference solver for nonlinear two-point boundary problems with mild boundary layers. *SIAM J. Numer. Anal.*, 14: 91–111

[101] R. J. LeVeque. 1992. *Numerical Methods for Conservation Laws*. Birkhauser, Basel, 2nd edition

[102] A. L. Levin and D. S. Lubinsky. 1992. Christoffel functions, orthogonal polynomials, and Nevai's conjecture for Freud weights. *Constr. Approx.*, 8: 461–533

[103] W. B. Liu and J. Shen. 1996. A new efficient spectral-Galerkin method for singular perturbation problems. *J. Sci. Comput.*, 11: 411–437

[104] W.-B. Liu and T. Tang. 2001. Error analysis for a Galerkin-spectral method with coordinate transformation for solving singularly perturbed problems. *Appl. Numer. Math.*, 38: 315–345

[105] Y. Liu, L. Liu, and T. Tang. 1994. The numerical computation of connecting orbits in dynamical systems: a rational spectral approach. *J. Comput. Phys.*, 111: 373–380

[106] J. M. Lopez. 1990. Axisymmetric vortex breakdown. I. Confined swirling flow. *J. Fluid Mech.*, 221: 533–552

[107] J. M. Lopez. 1990. Axisymmetric vortex breakdown. Part 1. Confined swirling flow. *J. Fluid Mech.*, 221: 533–552

[108] J. M. Lopez, F. Marques, and J. Shen. 2002. An efficient spectral-projection method for the Navier-Stokes equations in cylindrical geometries. II. Three-dimensional cases. *J. Comput. Phys.*, 176(2): 384–401

[109] J. M. Lopez and A. D. Perry. 1992. Axisymmetric vortex breakdown. III. Onset of periodic flow and chaotic advection. *J. Fluid Mech.*, 234: 449–471

[110] J. M. Lopez and A. D. Perry. 1992. Axisymmetric vortex breakdown. Part 3. Onset of periodic flow and chaotic advection. *J. Fluid Mech.*, 234: 449–471

[111] J. M. Lopez and J. Shen. 1998. An efficient spectral-projection method for the Navier-Stokes equations in cylindrical geometries I. axisymmetric cases. *J. Comput. Phys.*, 139: 308–326

[112] R. E. Lynch, J. R. Rice, and D. H. Thomas. 1964. Direct solution of partial differential equations by tensor product methods. *Numer. Math.*, 6: 185–199

[113] H.-P. Ma, W.-W. Sun, and T. Tang. 2005. Hermite spectral methods with a time-dependent scaling for second-order differential equations. *SIAM J. Numer. Anal.*

[114] Y. Maday, D. Meiron, A. T. Patera, and E. M. Rønquist. 1993. Analysis of iterative methods for the steady and unsteady Stokes problem: application to spectral element discretizations. *SIAM J. Sci. Comput.*, 14(2): 310–337

[115] Y. Maday, S. M. Ould Kaber, and E. Tadmor. 1993. Legendre pseudospectral viscosity method for nonlinear conservation laws. *SIAM J. Numer. Anal.*, 30(2): 321–342

[116] Y. Maday and T. Patera. 1989. Spectral-element methods for the incompressible Navier-Stokes equations. In A. K. Noor, editor, *State-of-the-art Surveys in Computational Mechanics*, pages 71–143

[117] Y. Maday, B. Pernaud-Thomas, and H. Vandeven. 1985. Reappraisal of Laguerre type spectral methods. *La Recherche Aerospatiale*, 6: 13–35

[118] Y. Maday and A. Quarteroni. 1988. Error analysis for spectral approximations to the Korteweg de Vries equation. *MMAN*, 22: 539–569

[119] G.I. Marchuk. 1974. *Numerical Methods in Numerical Weather Prediction*. Academic Press, New York

[120] M. Marion and R. Temam. 1998. Navier-Stokes equations: theory and approximation. In *Handbook of numerical analysis, Vol. VI*, Handb. Numer. Anal., VI, pages 503–688. North-Holland, Amsterdam

[121] G. Mastroianni and D. Occorsio. 2001. Lagrange interpolation at Laguerre zeros in some weighted uniform spaces. *Acta Math. Hungar.*, 91(1-2): 27–52

[122] R. M. M. Mattheij and G. W. Staarink. 1984. An efficient algorithm for solving general linear two point bvp. *SIAM J. Sci. Stat. Comput.*, 5: 745–763

[123] M. J. Mohlenkamp. 1997. *A Fast Transform for Spherical Harmonics*. PhD thesis, Yale University

[124] S. A. Orszag. 1970. Transform method for calculation of vector coupled sums: Applications to the spectral form of the vorticity equation. *J. Atmos. Sci.*, 27: 890–895

[125] S. A. Orszag. 1980. Spectral methods for complex geometries. *J. Comput. Phys.*, 37: 70–92

[126] S. A. Orszag. 1986. Fast eigenfunction transforms. *Science and Computers: Advances in mathematics supplementary studies*, 10: 23–30

[127] S. V. Parter. 2001. Preconditioning Legendre spectral collocation methods for elliptic problems. II. Finite element operators. *SIAM J. Numer. Anal.*, 39(1): 348–362

[128] A. T. Patera. 1986. Fast direct Poisson solvers for high-order finite element discretizations in rectangularly decomposable domains. *J. Comput. Phys.*, 65: 474–480

[129] L.P. Pitaevskii. 1961. Vortex lines in an imperfect Bose gase. *Sov. Phys. JETP*, 13: 451–454

[130] L. Quartapelle. 1993. *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhauser

[131] R. D. Richtmyper and K. W. Morton. 1967. *Difference Methods for Initial-Value Problems*. Interscience, New York, 2nd edition

[132] E. M. Ronquist. 1988. Optimal spectral element methods for the unsteady three-dimensional incompressible Navier-Stokes equations. *MIT Ph.D thesis*

[133] H. G. Roos, M. Stynes, and L. Tobiska. 1996. *Numerical Methods for Singularly Perturbed Differential Equations*. Springer Series in Computational Mathematics. Springer-Verlag, New York

[134] Y. Saad. 2003. *Iterative methods for sparse linear systems*. SIAM Philadelphia, PA, and edition

[135] Y. Saad and M. Schultz. 1986. Gmres: A genralized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7: 856–869

[136] W. W. Schultz, Lee, and J. P. Boyd. 1989. Chebyshev pseudospectral method of viscous flows with corner singularities. *J. Sci. Comput.*, 4: 1–24

[137] J. W. Schumer and J. P. Holloway. 1998. Vlasov simulations using velocity-scaled Hermite representations. *J. Comput. Phys.*, 144(2): 626–661

[138] J. Shen. 1991. Hopf bifurcation of the unsteady regularized driven cavity flows. *J. Comput. Phys.*, 95: 228–245

[139] J. Shen. 1995. Efficient spectral-Galerkin method II. direct solvers for second- and fourth-order equations by using Chebyshev polynomials. *SIAM J. Sci. Comput.*, 16: 74–87

[140] J. Shen. 1995. On fast Poisson solver, inf-sup constant and iterative Stokes solver by Legendre Galerkin method. *J. Comput. Phys.*, 116: 184–188

[141] J. Shen. 1996. Efficient Chebyshev-Legendre Galerkin methods for elliptic problems. In A. V. Ilin and R. Scott, editors, *Proceedings of ICOSAHOM'95*, pages 233–240. Houston J. Math.

[142] J. Shen. 1997. Efficient spectral-Galerkin methods III. polar and cylindrical geometries. *SIAM J. Sci. Comput.*, 18: 1583–1604

[143] J. Shen. 2000. A new fast Chebyshev-Fourier algorithm for the Poisson-type equations in polar geometries. *Appl. Numer. Math.*, 33: 183–190

[144] J. Shen. 2000. Stable and efficient spectral methods in unbounded domains using Laguerre functions. *SIAM J. Numer. Anal*. 38: 1113–1133

[145] J. Shen. 2003. A new dual-Petrov-Galerkin method for third and higher odd-order differential equations: application to the KDV equation. *SIAM J. Numer. Anal.*, 41: 1595–1619

[146] J. Shen, T. Tang, and L. Wang. *Spectral Methods: Algorithms, Analysis and Applications*. in preparation.

[147] S. J. Sherwin and G. E. Karniadakis. 1995. A triangular spectral element method; applications to the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 123(1-4): 189–229

[148] C.-W. Shu and S. Osher. 1988. Efficient implementation of essentially non-oscillatary shock capturing schemes. *J. Comput. Phys.*, 77: 439–471

[149] C.-W. Shu and S. Osher. 1989. Efficient implementation of essentially non-oscillatary shock-wave schemes, II. *J. Comput. Phys.*, 83: 32–78

[150] A. Solomonoff. 1992. A fast algorithm for spectral differentiation. *J. Comput. Phys.*, 98: 174–177

[151] W.F. Spotz and G.F. Carey. 1998. Iterative and parallel performance of high-order compact systems. *SIAM J. Sci. Comput.*, 19: 1–14

[152] F. Stenger. 1993. *Numerical Methods Based on Sinc and Analytic Functions*. Springer-Verlag, New York, NY

[153] J. Strain. 1994. Fast spectrally-accurate solution of variable-coefficient elliptic problems. *Proc. Amer. Math. Soc.*, 122: 843–850

[154] G. Strang. 1968. On the construction and comparison of difference schemes. *SIAM J. Numer. Anal.*, 5: 506–517

[155] G. Szegö. 1975. *Orthogonal Polynomials*, volume 23. AMS Coll. Publ, 4th edition

[156] E. Tadmor. 1986. The exponential accuracy of Fourier and Chebyshev differencing methods. *SIAM J. Numer. Anal.*, 23(1): 1–10

[157] E. Tadmor. 1987. The numerical viscosity of entropy stable schemes for systems of conservation laws. I. *Math. Comp.*, 49(179): 91–103

[158] T. Tang. 1993. The Hermite spectral method for Gaussian-type functions. *SIAM J. Sci. Comput.*, 14: 594–606

[159] T. Tang and Z.-H. Teng. 1995. Error bounds for fractional step methods for conservation laws with source terms. *SIAM J. Numer. Anal.*, 32: 110–127

[160] T. Tang and M. R. Trummer. 1996. Boundary layer resolving pseudospectral methods for singular perturbation problems. *SIAM J. Sci. Comput.*, 17: 430–438

[161] R. Temam. 1969. Sur l'approximation de la solution des équations de Navier-Stokes par la méthode des pas fractionnaires II. *Arch. Rat. Mech. Anal.*, 33: 377–385

[162] R. Temam. 1984. *Navier-Stokes Equations: Theory and Numerical Analysis*. North-Holland, Amsterdam

[163] L. J. P. Timmermans, P. D. Minev, and F. N. Van De Vosse. 1996. An approximate projection scheme for incompressible flow using spectral elements. *Int. J. Numer. Methods Fluids*, 22: 673–688

[164] L. N. Trefethen. 1988. Lax-stability vs. eigenvalue stability of spectral methods. In K. W. Morton and M. J. Baines, editors, *Numerical Methods for Fluid Dynamics* III, pages 237–253. Clarendon Press, Oxford

[165] L. N. Trefethen. 2000. *Spectral Methods in MATLAB*. SIAM, Philadelphia, PA

[166] H. Vandeven. 1991. Family of spectral filters for discontinuous problems. *J. Sci. Comput.*, 8: 159–192

[167] J. A. C. Weideman. 1992. The eigenvalues of Hermite and rational spectral differentiation matrices. *Numer. Math.*, 61: 409–431

[168] J. A. C. Weideman and S. C. Reddy. 2000. A matlab differentiation matrix suite. *ACM Transactions on Mathematical Software*, 26: 465–511

[169] J. A. C. Weideman and L. N. Trefethen. 1988. The eigenvalues of second-order spectral differentiation matrices. *SIAM J. Numer. Anal.*, 25: 1279–1298

[170] B. D. Welfert. 1997. Generation of pseudospectral differentiation matrices. *SIAM J. Numer. Anal.*, 34(4): 1640–1657

[171] N.N. Yanenko. 1971. *The Method of Fractional Steps*. Springer-Verlag, New York

[172] H. Yoshida. 1990. Construction of higher order symplectic integrators. *Phys. Lett. A.*, 150: 262–268

[173] J. Zhang. 1997. *Multigrid Acceleration Techniques and Applications to the Numerical Solution of Partial Differential Equations*. PhD thesis, The George Washington University `http://cs.engr.uky.edu/~jzhang/pub/dissind/html`

Index

A

Adams-Bashforth method, 43, 44
Advection equation, 256, 257

B

BiCGM, 48, 50
BiCGSTAB method, 48, 53, 54, 60, 122
Boundary layer, 91, 92, 98, 183–186, 293, 295
Burgers' equation, 152, 196, 200–202, 214–216, 221

C

Cauchy-Schwarz inequality, 65, 133, 134, 137, 138, 141, 181, 281
Chebyshev
  coefficients, 116
  collocation method, 68, 84, 86, 91, 92, 98, 102, 126, 132, 133, 136, 223, 234, 235, 257, 259, 260, 262, 263, 310, 311
  derivative matrix, 301
  differentiation matrix, 259, 260, 301, 307
  expansions, 6, 119, 233
  interpolant, 300
  polynomial, 2, 15–18, 21–23, 61, 70, 113, 115, 118, 119, 122, 125, 170, 172–174, 192, 196, 240, 244, 245, 291, 292, 308, 310, 312
  spectral method, 2, 18, 22, 167, 183, 196, 203
  transform, 116–118, 239
    discrete, 15, 17, 18, 113
Chebyshev Gauss points, 244
Chebyshev Gauss-Lobatto points, 18, 66, 67, 74, 77, 87, 88, 113, 114, 118, 119, 197, 234, 236, 237, 259, 260
Condition number, 49, 58, 68, 85, 87–89, 91, 104, 121, 122, 124, 126
Conjugate gradient method, 48, 49, 58, 125, 126, 277, 280
Conjugate gradient squared, 48, 51, 122, 125
Coordinate transformation, 6, 152, 244, 292, 300
Differentiation matrix, 5, 6, 46, 69, 73, 74, 76, 78, 79, 81, 82, 84, 85, 93, 148, 154, 163, 168, 234, 236, 299, 300, 309
  Chebyshev, 259, 260, 301
  Fourier, 259, 301
  Hermite, 301
  Laguerre, 301
  sinc, 301

E

Explicit scheme, 41, 85, 91, 262, 264

F

Fast Cosine Transform, 27, 34, 259, 310
Fast Fourier Transform, 1, 3, 5, 18, 27, 28, 31, 33–35, 37, 84, 113,

116–119, 125, 195, 199, 206, 247, 249, 253, 310
Filter
exponential, 217, 259, 262
Lanczos, 217
raised cosine, 217
sharpened raised cosine, 217
spectral, 183, 214, 217, 218
Filtering, 218, 223, 226–228, 257, 300, 310, 311
Finite-difference method, 3, 4, 69, 97, 185, 200, 247, 249, 293, 299, 309
Finite-element method, 2–4, 109, 282, 287, 293, 299, 309
Fourier
coefficients, 216, 218–220, 222–225, 228
collocation method, 79, 84, 223, 257, 259, 262, 263, 308, 310
differentiation matrix, 259, 301
expansion, 233
series, 2, 79, 80, 143, 170, 216, 309
sine transform, 247–249, 310
spectral method, 2, 4, 81, 82, 183, 204, 214, 223
transform
continuous, 36
discrete, 27, 36, 206, 207

G

Gauss points, 267
Gauss-Lobatto points, 65, 66, 91, 107, 126, 239, 293
Gauss-Radau points, 271

H

Heat equation, 2, 5, 6, 46, 85, 153, 154, 196–198
Helmholtz equation, 290, 291
Hermite polynomials, 70, 266, 302, 303, 306
Hermite-Gauss points, 148, 149, 267

I

Inner product, 7, 22, 61, 80, 112, 124, 125, 133, 146, 152, 181, 237, 244, 285
discrete, 6, 14, 100, 103, 136

J

Jacobi polynomial, 23–26, 61, 64, 131, 138, 143, 300

L

Lagrange interpolation polynomial, 71, 76, 92, 136
Lagrange polynomial, 70, 75, 78, 83, 105, 190, 192, 234
Laguerre Gauss-Lobatto points, 167, 168
Laguerre polynomial, 143, 158, 159, 161, 162, 167, 178, 269, 302, 303, 306
Laguerre-Gauss points, 160
Laguerre-Gauss-Radau points, 160, 161, 163, 167, 168, 271
Laplace operator, 276, 289
Legendre
coefficients, 22, 112, 118, 128, 131
collocation method, 102, 104, 126, 234, 259, 263, 310
expansion, 118, 119, 297
polynomial, 2, 15, 18–20, 22, 23, 26, 61, 70, 78, 118, 119, 122,

127, 130, 240, 244, 245, 291, 292, 302, 303, 308, 310, 312
spectral method, 2, 22, 167
spectral-Galerkin method, 283
transform, 112, 118, 120, 128, 131, 239
discrete, 15, 22, 23, 113, 118
Legendre Gauss points, 244
Legendre Gauss-Lobatto points, 21, 22, 78, 79, 112, 113, 127, 128, 130, 131, 234, 237
Legendre Gauss-Radau points, 23

M

Matrix diagonalization method, 237
Multistep method, 38, 42, 44

N

Navier-Stokes equations, 241, 282, 284, 287, 289
Neumann boundary condition, 84, 88, 89, 110, 115, 237, 240, 241, 285

O

Orthogonal polynomials, 1, 6, 7, 9, 10, 12–15, 20, 23, 25, 105, 109, 143, 301, 303
Orthogonal projection, 61, 62, 64, 132, 135, 138, 177–179, 181, 182
Orthogonality property, 215, 216
Chebyshev, 119
Legendre, 119, 127, 130

P

Poisson equation, 182, 233, 235, 237, 243, 250, 269, 284, 285, 287, 290
Preconditioning, 48, 58, 121, 122, 125
conjugate gradient, 58
finite difference, 99, 101
finite element, 99, 102
GMRES, 59
iterative method, 122
Projection method, 241
Pseudospectral methods, 68, 106, 107, 172, 173, 184, 185, 190, 193, 264, 266, 271–274

Q

Quadrature rule, 162
Gauss type, 12–14, 17, 147, 150, 159, 267, 310
Chebyshev, 18
Laguerre, 159
Gauss-Lobatto type, 13, 310
Chebyshev, 17, 100, 136
Legendre, 22, 100, 124
Gauss-Radau type, 13, 17, 310
Laguerre, 160–163
Hermite-Gauss type, 145, 146, 148

R

Recurrence relations, 9, 16, 18, 20, 22, 23, 25, 26, 51, 52, 120, 127, 144, 146, 148, 159, 161, 163, 303
Reynolds number, 6, 288
Robin condition, 301
Runge-Kutta scheme, 38, 39, 41, 44, 203, 204, 216, 218, 219, 229, 259, 262
RK2, 39, 46
RK3, 40, 216, 220
RK4, 40, 41, 47, 204, 206, 209
stability, 41, 42

S

Scaling factor, 144, 150, 152, 153, 155, 157, 158, 167, 168, 170, 175
Shocks, 222, 223, 225, 226
Sobolev inequality, 178
Sobolev space, 7, 61, 66, 181
Spectral accuracy, 157, 171, 176, 222, 223, 228, 260, 299, 309
Spectral projection method, 282, 287, 288, 295
Spectral radius, 68, 85–89, 91, 261
Splitting
  error, 266, 285
  method, 38, 45, 265, 266, 282–287
Stokes flow, 294, 295
Stokes problem, 295

U

Uzawa
  algorithm, 276, 278–281
  operator, 276, 278