

Problem Books in Mathematics

Tom Lyche
Jean-Louis Merrien

Exercises in Computational Mathematics with MATLAB

 Springer

Problem Books in Mathematics

Series Editor:

Peter Winkler

Department of Mathematics

Dartmouth College

Hanover, NH 03755

USA

For further volumes:

<http://www.springer.com/series/714>

Tom Lyche • Jean-Louis Merrien

Exercises in Computational Mathematics with MATLAB



Springer

Tom Lyche
University of Oslo
Department of Mathematics
Oslo
Norway

Jean-Louis Merrien
INSA de Rennes CS 70839
Rennes Cedex 07
France

ISSN 0941-3502

ISBN 978-3-662-43510-6

ISBN 978-3-662-43511-3 (eBook)

DOI 10.1007/978-3-662-43511-3

Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014948729

Mathematics Subject Classification (2010): 65-D, 65-F, 65-H, 65-L, 68-W

© Springer-Verlag Berlin Heidelberg 2014

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This book was written for advanced undergraduate and beginning graduate students who need a background in numerical analysis and computational science, including students of mathematics, statistics, computational physics and engineering. Mastering the material in this book will enable students to become skilled in numerical analysis and in MATLAB programming.

There are many excellent textbooks on numerical analysis, scientific computing and MATLAB programming, and we include some of them in the bibliography. However, this book is structured differently. While most of the concepts presented are not new, we propose studying the topics by solving exercises. Our goal is to give students a tool to enable them to work independently. Most chapters begin with a review followed by a selection of “theoretical” and programming exercises. Some of the exercises combine both theory and programming, such as a theoretical error bound and its numerical computation. To help beginning students, the exercises in MATLAB programming are often presented in the form of “Russian dolls.” That is, each new question improves on and completes the previous program, and results are provided to validate the intermediate programs. At the end of each chapter, complete solutions and programs are provided.

We feel that this approach, which links theory and programming, is essential to better understanding of numerical analysis and scientific computing. Also, improving the creation of programs requires some background knowledge.

Four tools are proposed to improve and facilitate exercises and understanding:

- A paragraph **Review:** ...
- Footnotes *>Math Hint¹<*,
- Footnotes *< MATLAB Hint²>*,
- A rubric **Comment:**... ☺

giving further comments.

At the end of the book, three reference indices are also provided:

¹ Help for the mathematical part of a problem

² Help with the programming in MATLAB

- An index of mathematicians and scientists who have made important contributions to the field. Short biographies are included in the different chapters so that the reader can see how numerical analysis and computational science fit into the history and geography of science. Most of these biographies were prepared with the help of Wikipedia.
- A standard index to help readers find where and how some concepts are used.
- An index of MATLAB commands to improve technical programming skills.

The first chapters can be used as an introduction and to learn a number of basic tools. Here is a brief overview of the book's content.

Chapter 1 informally reviews useful commands in MATLAB for dealing with tables, vectors and matrices and provides basic commands for plotting. These essential MATLAB commands and functions are the basis for progressing in programming. The reader should not hesitate to frequently use the online help.

In the second, third and fourth chapters, the central notion of matrices in numerical analysis is addressed. Many numerical methods involve a linear system that has to be solved. As such, we present a few of the methods for solving linear systems in Chap. 2 and many more are included at regular intervals throughout the book. They can easily be located using the index.

Eigenvalues and eigenvectors are studied in Chap. 3. In this chapter we solve some problems that can be formulated as eigenvalue problems and the basis of eigenvectors (if it exists) is used to simplify a matrix or a linear transformation. Perturbation theory provides information on the difficulties involved in obtaining accurate numerical solutions to mathematical problems. In Chap. 4 we study the most important vector and matrix norms and use them to study the effect of perturbing elements in a linear system. The condition numbers of linear systems play an important role in this analysis, and several examples are studied.

In Chapter 5, we examine different iterative methods to solve nonlinear equations in the fixed point form $x = f(x)$. Iterative methods for linear systems are also addressed. We have added an exercise using subdivision to create interpolating curves; this subdivision tool is a relatively recent concept in mathematics (from the late 1980s).

The next three chapters deal with polynomials and piecewise polynomials and how they can be used to construct curves and surfaces. Interpolation is studied in Chap. 6. The purpose is to find a function that matches the given data. After preliminary practice exercises, we present a more complete study of polynomial Lagrange interpolation. With suitable hypotheses, the solution is unique, but it can be computed using different bases. In the last part students plot surfaces in space using interpolation of data on a rectangular grid.

Bézier techniques have emerged as a relatively recent tool in mathematics. Chapter 7 provides an introduction to the construction of Bézier curves and surfaces in computer-aided design, which are used to produce medical images, to mention just one application.

In Chap. 8 we take a second look at approximation processes. Runge's phenomenon shows that the choice of interpolation points can be important. Moreover, instead of using polynomials of high degree, better approximations can usually be

obtained by using piecewise polynomials or splines. We also apply splines to compute an approximation of the length of a curve.

Most integrals cannot be precisely computed and many numerical methods are known, some of which are studied in Chap. 9 with the construction of elementary quadratures, composite rules, extrapolation to the limit and a study of the errors. All of these methods can also be used to approximate the solutions of integral equations, and examples are provided.

Instead of finding a function using interpolation, the next two chapters focus on the question of approximating data. In Chap. 10, we look for least-squares approximations. Examples from signal processing are presented. The essential tool here is orthogonality, and details are provided to show how it works. Applications using different classes of functions are proposed, e.g., using trigonometric functions rather than polynomials to approximate periodic signals. The question of approximation is also studied in Chap. 11, where we consider polynomial approximation using the infinity norm, Fourier series solution of the heat equation and Haar wavelets.

In the final two chapters, we study numerical methods for differential equations. Chapter 12 considers ordinary differential equations. We consider some explicit and implicit Runge-Kutta methods and solve two point boundary value problems using the shooting method. Finally, Chap. 13 deals with finite difference methods for partial differential equations.

As problems involving nonlinear equations often occur in scientific computing, readers can find exercises on them in Chaps. 5, 12, and 13. We also treat least-squares methods, but have decided not to deal with optimization methods, since this would have required several extra chapters.

Many of the exercises have been tested by students at the University of Oslo and at INSA Rennes. Some MATLAB exercises have been taken from examination questions, and can be mastered in less than two hours with a little practice.

Oslo, Norway
Rennes, France
April 2014

Tom Lyche
Jean-Louis Merrien

Contents

1	An Introduction to MATLAB Commands	1
1.1	Elementary Commands	1
1.2	Matrices, Vectors, Arrays	2
1.3	M-Files	4
1.4	Mathematics and MATLAB	5
1.5	Examples of Elementary Graphs	6
1.6	Solutions for Sect. 1.4	8
2	Matrices and Linear Systems	9
2.1	Some Matrix Computations	10
2.2	Block Multiplication	12
2.3	Diagonally Dominant Matrices	13
2.4	LU-Factorization of Diagonally Dominant Tridiagonal Systems	15
2.5	Symmetric Positive Definite Systems and Cholesky Factorization	16
2.6	Solutions	16
2.6.1	Some Matrix Computations	16
2.6.2	Block Multiplication	19
2.6.3	Diagonally Dominant Matrices	21
2.6.4	LU-Factorization of Diagonally Dominant Tridiagonal Systems	21
2.6.5	Symmetric Positive Definite Systems and Cholesky Factorization	22
3	Matrices, Eigenvalues and Eigenvectors	25
3.1	Elementary Computations	26
3.2	The Power and Inverse Power Methods	28
3.3	The QR Method	30
3.4	Application to the Buckling of a Beam	31
3.5	Solutions	33
3.5.1	Elementary Computations	33
3.5.2	The Power and Inverse Power Methods	36

3.5.3	The QR Method	38
3.5.4	Application to the Buckling of a Beam	40
4	Matrices, Norms and Conditioning	43
4.1	Elementary Examples	44
4.2	Conditioning and Error	45
4.3	Conditioning, Eigenvalues and Determinant	47
4.4	Solutions	54
4.4.1	Elementary Examples	54
4.4.2	Conditioning and Error	55
4.4.3	Conditioning, Eigenvalues and Determinant	57
5	Iterative Methods	65
5.1	Fixed-Point Methods	66
5.2	Iterative Methods for Linear Systems	69
5.2.1	The Methods of Jacobi and Gauss Seidel	69
5.2.2	Gradient Methods	72
5.3	Subdivision Schemes	76
5.4	A Nonlinear Pendulum	78
5.5	Solutions	83
5.5.1	Fixed Point Methods	83
5.5.2	Iterative Methods for Linear Systems	86
5.5.3	Subdivision Schemes	94
5.5.4	A Nonlinear Pendulum	97
6	Polynomial Interpolation	103
6.1	Some Special Cases	105
6.2	Lagrange Interpolation	106
6.2.1	The Lagrange Basis	107
6.2.2	The Newton Form	107
6.2.3	The Error Term	108
6.2.4	The Runge Phenomenon	110
6.3	Stable Lagrange Interpolation	111
6.3.1	The Univariate Case	111
6.3.2	Application to Interpolating Surfaces	114
6.4	Approximation of Derivatives, Part I	115
6.5	Solutions	118
6.5.1	Some Special Cases	118
6.5.2	Lagrange Interpolation	121
6.5.3	Stable Lagrange Interpolation	125
6.5.4	Approximation of Derivatives, Part I	128
7	Bézier Curves and Bernstein Polynomials	131
7.1	Bézier Curves and the de Casteljau Algorithm	132
7.2	Bernstein Polynomials	136
7.3	Shape Preservation	140

7.4	Smooth Joining of Bézier Curves	141
7.5	Solutions	142
7.5.1	Bézier Curves and the de Casteljau Algorithm	142
7.5.2	Bernstein Basis Polynomials	144
7.5.3	Shape Preservation	147
7.5.4	Smooth Joining of Bézier Curves	149
8	Piecewise Polynomials, Interpolation and Applications	153
8.1	Quadratic and Cubic Hermite Interpolation	154
8.2	Cubic Spline Interpolation	156
8.3	Approximation of the Derivatives	163
8.4	Approximation of the Length of a Curve	166
8.5	Solutions	167
8.5.1	Quadratic and Cubic Hermite Interpolation	167
8.5.2	Cubic Spline Interpolation	170
8.5.3	Approximation of the Derivatives	173
8.5.4	Approximation of the Length of a Curve	174
9	Approximation of Integrals	177
9.1	Basic Quadrature Rules	179
9.1.1	Quadrature Rules and Degree of Precision	179
9.1.2	Weighted Quadrature Rules	181
9.2	Monte Carlo Method	184
9.3	Trapezoidal Rule	186
9.4	Extrapolation	192
9.5	A Global Quadrature	195
9.6	Solutions	199
9.6.1	Basic Quadrature Rules	199
9.6.2	Monte Carlo Method	206
9.6.3	Trapezoidal Rule	208
9.6.4	Extrapolation	212
9.6.5	A Global Quadrature	216
10	Linear Least Squares Methods	221
10.1	Orthogonal Projections and Orthogonal Sums	222
10.2	Curve Fitting by Least Squares	225
10.3	Periodic Signal	229
10.4	Savitsky-Golay Filter	232
10.5	Ordinary Differential Equation and Least Squares	236
10.6	Solutions	238
10.6.1	Orthogonal Projections and Orthogonal Sums	238
10.6.2	Curve Fitting by Least Squares	241
10.6.3	Periodic Signal	244
10.6.4	Savitsky-Golay Filter	245
10.6.5	Ordinary Differential Equation and Least Square	247

11 Continuous and Discrete Approximations	249
11.1 Polynomial Approximation with $\ \cdot\ _\infty$	249
11.2 Fourier Series Solution of the Heat Equation	258
11.3 Discrete Signal and Haar Wavelets	261
11.4 Solutions	266
11.4.1 Polynomial Approximation with $\ \cdot\ _\infty$	266
11.4.2 Fourier Series Solution of the Heat Equation	274
11.4.3 Discrete Signal and Haar Wavelets	276
12 Ordinary Differential Equations, One Step Methods	281
12.1 Linear Differential System and Euler's Methods	285
12.2 Other Examples of One Step Methods	288
12.3 Predictor-Corrector	293
12.4 Application of ODE Solvers to the Shooting Method	294
12.5 Solutions	301
12.5.1 Linear Differential System and Euler's Methods	301
12.5.2 Other Examples of One Step Methods	305
12.5.3 Predictor-Corrector	313
12.5.4 Application of ODE Solvers to the Shooting Method	314
13 Finite Differences for Differential and Partial Differential Equations	321
13.1 Definitions of Finite Differences	321
13.2 Applications of Finite Differences in Dimension 1	323
13.3 Finite Differences in Dimension 2	332
13.4 The Heat Equation and Approximations	337
13.5 Solutions	342
13.5.1 Definitions of Finite Differences	342
13.5.2 Applications of Finite Differences in Dimension 1	344
13.5.3 Finite Differences in Dimension 2	352
13.5.4 The Heat Equation and Approximations	357
References	363
Index of Names	365
Subject Index	367
MATLAB Index	371

Chapter 1

An Introduction to MATLAB Commands

The MATLAB language is a high-level matrix/array language that also allows object oriented language features. In this chapter, we recall some simple MATLAB commands. This is not a complete description of MATLAB, we only propose a few commands that can be useful for programming numerical algorithms. MATLAB commands are written in boldface.

This chapter will give you indispensable tools and some tricks to start programming in MATLAB. We first give some elementary commands and then focus on arrays and matrices. MATLAB programs are written and stored in so called M-files. A few examples of such files are given. To reduce the computation time we will use MATLAB tools for vector and matrix calculations in order to avoid for loops of the type **for** $i=1:n$. This develops good programming habits that can be important when writing programs for scientific and industrial applications. MATLAB uses floating point arithmetic and some constants describing the precision of MATLAB is given in Sect. 1.4. Here we also give an exercise showing that the accuracy of a computed result can depend on how the underlying mathematical problem is formulated. We end with some useful graphic commands both for plotting curves and surfaces.

1.1 Elementary Commands

Test these elementary examples:

Numerical operations

```
>> 3*7, 3^3  
>> 1+3*2^5
```

How to create new variables

```
>> x=2
>> y=x^5
>> y/x, z=y/x^2;
>> t=input('Enter a real number: ')
```

Test: Guess the output before typing

```
>> x=2;
>> x^3,x^4;
>> -1+2*(3+x)
>> z=-1+2*3+x;
>> z
```

Format of the variables:

```
>> a=sqrt(3)
>> format long , b=sqrt(3)
>> a-b
>> format short
>> who
>> clear
>> who
```

⌚Comment: The list of the variables are also given in the Workspace browser. There, you can check your variables and their dimensions. This is sometimes useful when debugging programs. You can also modify the value of a variable. ☺

1.2 Matrices, Vectors, Arrays

One important point is that MATLAB is a matrix/array language; hence, a single number is a 1×1 array or matrix. Every variable is an array and MATLAB has special functions for the usual and not so common matrix operations.

How to create or modify a matrix

```
>> a=[1,2,3;-6 5 -4]
>> a(1,2), a(2,3), a(4,5)
>> a(2,3)=10
>> a'
>> rand(1,3), rand(2)
>> zeros(3), zeros(3,3)
>> ones(3,2)
>> eye(3), eye(2,3)
>> magic(3), sum(magic(3)), sum(sum(magic(3)))
>> help magic
>> b=[1 4 5], diag(b)
>> c=diag(a)
>> whos
```

⌚Comment: Instead of using `>> help magic`, you can use the online help. ☺

You have seen that a command can be finish by a semicolon and this cancels the output. The semicolon is essential when generating large matrices.

```
>> s1=zeros(20,25) ; s2=zeros(2,3)
>> s3=zeros(100)
```

The operator “:” is useful for constructing arrays and to extract part of an array, a row or a column.

```
>> -3:3
>> x=-3:0.3:3
>> x(2:12)
>> x(9:-2:1)
>> x=10:100; x(2), x(10)
>> x(40:5:60)
>> a=[1:6 ; 2:7 ; 4:9]
>> a(1, :), a(:, 2)
>> s=rand(20,5); s(6:7, 2:4)
>> a=[1,2 ; 3,4]
>> sin(a*pi/4)
```

Test: Guess the output before typing the commands without semicolon

```
>> x=-1:2:5;
>> x(2), x(8)
>> x(3:4), x(1:6)
>> diag(x), diag(x,-1)
>> a=2*eye(3)
>> a(:,2), a(1,:),
>> a(2:3,:)
>> diag(a)
```

Matrix computations

```
>> a=[1 2 3; 4 5 6; 7 8 10] , b=[1 1 1]'
>> 2*a , a/4
>> a*b
>> b*a
>> b'*a
>> b'*b , b*b'
>> a^2
>> a(:)
>> c=b', b, b(:), c(:)
```

Mixing scalars and vectors

```
>> a+1, b+2
>> a+[b,b,b]
```

Dot operations

```
>> a^2, a.^2, a.*a
>> a.*b
>> 1./a, 1./a.^2
```

Test: Guess the value of z before typing the following commands

```
>> a=[3 2 1 ; 2 3 2 ; 1 2 3];
>> x= a(:,1).*a(:,3);
>> y=a(2,:);
>> b=x*y;
>> z=[b(1,:); b(2,:); b(3,:)];
>> z(1:2:9)=ones(1,5)
```

Linear systems

The linear system $ax = b$ can be solved with a unique instruction. Test:

```
>> x=a\b
```

Check when computing: $>> a*x$, $a*x-b$.

You can also solve the system $ya = b'$ with $>> y=b' / a$. Try also $>> y=b/a$.

Some other matrix functions

```
>> det(a), rank(a), inv(a), eig(a)
```

1.3 M-Files

Script M-file

You can also write some of the previous computations in an M-file. Open a new M-file (menu file on the left), type your commands

```
x=2;
x^3,y=x^4;
disp('Here is y: '),y
-1+2*(3+x)
z=-1+2*3+x;
z
a=2*eye(3)
```

Change directory to be in your personal environment. Give a name to your file while registering (for example `calculus1.m`), then in the MATLAB command window, type `>> calculus1`

M-Function

Now we would like to compute the function $f(x) = \sqrt{1+x}$ for an array X of different values of x . Open a new M-file (menu file on the left), type

```
function y=f1(x)
y=sqrt(1+x);
```

A function should be saved in a file having the same name as the function, in this case `f1.m`. Then in the MATLAB command window, type

```
>> X=0:0.5:3;
>> Y=f1(X)
```

```
>> f1(1:4)
>> feval('f1',1:4)
```

Loop or no loop?

Construct three M-files `loop1.m`, `loop2.m` and `direct.m` with the instructions

`loop1.m:`

```
for i=1:2000
    for j=1:2000
        t1(i,j)=i/j;
    end
end
```

`loop2.m:`

```
t2=zeros(10000);
for i=1:10000
    for j=1:10000
        t2(i,j)=i/j;
    end
end
```

`direct.m:`

```
i=1:10000;
j=1./(1:10000);
t3=i.*j;
```

You can time the three computations as follows:

```
>> tic;loop1;toc
>> tic;loop2;toc
>> tic; direct;toc
```

If we had used 10000 instead of 2000 in `loop1.m` then the three computations would, apart from rounding errors, give $t_1=t_2=t_3$. You can check if $t_1=t_2$ with the command (`>> max(max(abs(t2-t1)))`)., MATLAB has adapted operators for vectorized computations and you should make an effort to find a fast way to do the computation.

1.4 Mathematics and MATLAB

MATLAB and real numbers

What is `eps`? Discover details when typing (`>> help eps`) or in the online help.

```
>> eps
>> 1+eps-1
>> 1+eps/2-1
```

```
>> eps/3
>> 1e-16+eps/2
>> realmin
>> realmin/2
>> realmax
```

Comment: $1+\text{eps}$ is the smallest number greater than 1 on your computer. which means that eps gives the best relative precision that you can get using MATLAB. realmin is the smallest positive normalized floating-point number on your computer. realmax is the largest floating-point number representable on your computer.❷

Precision

Exercise 1.1. (Mathematically – but not numerically equivalent)

We are going to compute the function $f(x) = \frac{\sqrt{1+x}-1}{x}$ for different values of x in the neighborhood of 0. We first notice that $f(x) = \frac{1}{\sqrt{1+x}+1}$ and also $f(x) = 1/2 - 1/8x + 1/16x^2 - 5/128x^3 + o(x^3)$. Create three m-functions with

$$f_2(x) = \frac{\sqrt{1+x}-1}{x}, f_3(x) = \frac{1}{\sqrt{1+x}+1}, f_4(x) = 1/2 - 1/8x + 1/16x^2 - 5/128x^3.$$

Now, in the command window, create an array $X = [10^{-10}, 10^{-12}, 10^{-14}, 10^{-16}]$ then create an array A which first row is X , and successive rows are $f_2(X)$, $f_3(X)$, $f_4(X)$. Next, add the commands `>> format long` and then `>> format long e`.

1.5 Examples of Elementary Graphs

One variable functions

```
>> x = -10: .01:10 ;
>> plot(x.*x)
>> figure
>> plot( x , x.*x )
>> plot( x , x.*sin(x) )
>> plot(x.*cos(x),x.*sin(x) )
>> close all
>> x=-10: .001:10 ;
>> comet(x.*cos(x),x.*sin(x) )
>> title('Nice, isn''t it?');
```

Two variables functions

```
>> [x,y]=meshgrid ([0:1:3],[1:0.5:2])
>> [x y]=meshgrid (-3:0.1:3, -4:0.2:5);
>> z = x.^2 + y.^2;
```

```
>> mesh(x,y,z)
>> surf(x,y,z)
>> xlabel('x')
>> contour(x,y,z)
```

If you want to modify your figure, you can use the command `plottools`.

About meshgrid

Type the following commands:

```
>> X=[1,2,3],Y=[-1,1]
X =
    1      2      3
Y =
    -1      1
>> [XX1,YY1]=meshgrid(X,Y)
XX1 =
    1      2      3
    1      2      3
YY1 =
    -1      -1      -1
    1      1      1
>> [YY2,XX2]=meshgrid(Y,X)
YY2 =
    -1      1
    -1      1
    -1      1
XX2 =
    1      1
    2      2
    3      3
>> ZZ1=sin(XX1+YY1)
ZZ1 =
    0      0.8415      0.9093
    0.9093      0.1411     -0.7568
>> ZZ2=sin(XX2+YY2)
ZZ2 =
    0      0.9093
    0.8415      0.1411
    0.9093     -0.7568
```

Comment: If X, Y are arrays of reals with respective dimension p and q , $[XX1, YY1] = \text{meshgrid}(X, Y)$ creates arrays of dimension $q \times p$, while $[YY2, XX2] = \text{meshgrid}(Y, X)$ creates arrays of dimension $p \times q$.

Also, if $F.m$ is a file with a function f of 2 variables, in the example above $f(x, y) = \sin(x + y)$, then

>> ZZ1=ff(XX1,YY1) creates an array of dimension $q \times p$ which elements are $ZZ1(i, j)$ with values $f(x_j, y_i)$ while
>> ZZ2=ff(XX2,YY2) creates an array of dimension $p \times q$ which elements are $ZZ2(i, j)$ with values $f(x_i, y_j)$. ☺

1.6 Solutions for Sect. 1.4

Exercise 1.1

The functions

```
function y=f2(x)
y=(sqrt(1+x)-1)./x;

function y=f3(x)
y=1./(sqrt(1+x)+1);

function y=f4(x)
y=1/2-1/8*x+1/16*x.*x-5/128*x.^3;
```

Main program

```
clear
X=[1e-10,1e-12,1e-14,1e-16];
A=[X;f2(X);f3(X);f4(X)]
format long
A
format long e
A
```

Chapter 2

Matrices and Linear Systems

Many complex mathematical problems have a linear system of equations hidden in it. For example, to solve a nonlinear system of equations a linearization of the problem will lead to a sequence of linear systems. To give a command to solve a general linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ in MATLAB is simple. Most often $\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$ will do the trick. However, problems can occur. The matrix \mathbf{A} can be singular or almost singular and then the computed solution can be quite inaccurate. For a better understanding of when such problems can occur one needs some background in linear algebra.

We start with a very brief review of some linear algebra concepts related to linear systems while other reviews will be given at the beginning of the sections. A few exercises recalling some matrix concepts are proposed in the first section (Exercises 2.1–2.4). Block multiplication of matrices is an important tool and we devote the next section to such operations (Exercises 2.5–2.7). In the following sections, we exploit the structure of diagonally dominant matrices (Exercise 2.8), then tridiagonal matrices in Exercise 2.9. The final section is about symmetric positive definite matrices (Exercise 2.10) and Cholesky factorizations that often occur in applications.

Review: Consider a linear system

$$(S) \left\{ \begin{array}{l} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{array} \right.$$

of m equations in n unknowns. Every coefficients a_{ij} , the unknowns x_j , and the components of the right hand sides b_i are in \mathbb{R} or \mathbb{C} . The system can be written as a matrix equation

$$Ax = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = \mathbf{b}.$$

The system is **overdetermined** if $m > n$, **square** if $m = n$ and **underdetermined** if $m < n$. It is **homogenous** if $\mathbf{b} = \mathbf{0}$. A linear system may have a unique solution, infinitely many solutions, or no solution.

We recall

1. The **column space**, **null space**, **rank**, and **nullity** of a matrix $A \in \mathbb{R}^{m \times n}$ are defined by $\text{span}(A) := \{\mathbf{y} \in \mathbb{R}^m : \mathbf{y} = Ax, \mathbf{x} \in \mathbb{R}^n\}$, $\ker(A) := \{\mathbf{x} \in \mathbb{R}^n : Ax = \mathbf{0}\}$, $\text{rank}(A) := \dim(\text{span}(A))$, and $\text{null}(A) := \dim(\ker(A))$. Recall that $\text{rank}(A) + \text{null}(A) = n$.
2. A square matrix A is said to be **nonsingular** if the only solution of the homogenous system $Ax = \mathbf{0}$ is $x = \mathbf{0}$.
3. A square matrix $A \in \mathbb{R}^{n \times n}$ is nonsingular if and only if one of the following properties is satisfied:
 - a. The determinant $\det(A) = |A|$ is nonzero,
 - b. Any square linear system $Ax = \mathbf{b}$ has a unique solution,
 - c. $BA = AB = I$ for some square matrix $B \in \mathbb{R}^{n \times n}$. In this case, $B = A^{-1}$ is unique if it exists and is called the **inverse** of A ,
 - d. $\text{span}(A) = \mathbb{R}^n$,
 - e. $\text{rank}(A) = n$,
 - f. $\ker(A) = \{\mathbf{0}\}$,
 - g. $\text{null}(A) = 0$,
 - h. Zero is not an eigenvalue (see Chap. 3).

2.1 Some Matrix Computations

Exercise 2.1.

Let $A = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix}$. With MATLAB compute $\det A$.

Exercise 2.2.

Determine the rank of $A = \begin{bmatrix} 2 & 1 & 3 & 3 \\ 1 & 0 & -1 & 0 \\ a & 2 & 1 & 1 \\ 4 & 3 & 2 & 4 \end{bmatrix}$ as a function of a . Check with MATLAB and the function `rank` for different values of a .

Exercise 2.3.

Let $\mathbf{A} = \begin{bmatrix} 2 & 4 & 6 \\ 0 & 2 & 3 \\ 0 & 0 & 2 \end{bmatrix}$. We define $\mathbf{B} = \mathbf{A} - 2\mathbf{I}$.

1. Compute by hand \mathbf{B}^k for any $k \in \mathbb{N}$.
2. Find \mathbf{A}^n for $n \in \mathbb{N}$.
3. Compute $\left(\mathbf{I} + \frac{\mathbf{B}}{2}\right) \times \left(\mathbf{I} - \frac{\mathbf{B}}{2} + \frac{\mathbf{B}^2}{4}\right)$ and use the result to find \mathbf{A}^{-1} .
4. Find \mathbf{A}^{-n} for $n \in \mathbb{N}$.
5. We have $\mathbf{A}^n = F(n)$ and $\mathbf{A}^{-n} = F(-n)$, where $F : \mathbb{R} \rightarrow \mathbb{R}^{2 \times 2}$ is a matrix valued function given by

$$F(x) = 2^{x-1} \begin{bmatrix} 2 & 4x & 3x(x+1) \\ 0 & 2 & 3x \\ 0 & 0 & 2 \end{bmatrix}$$

If we define $\mathbf{A}^x := F(x)$ for all $x \in \mathbb{R}$, show that $\mathbf{A}^x \mathbf{A}^y = \mathbf{A}^{x+y}$.

6. With MATLAB, create a file `mat1.m` where for given n , \mathbf{A}^n and \mathbf{A}^{-n} are computed using the power notation in MATLAB. Test with $n = 5$.

```
>> mat1
n =
      5
A^(5) :
ans =
      32          320         1440
          0          32          240
          0            0          32
A^(-5) :
ans =
    0.0313   -0.3125    0.9375
        0     0.0313   -0.2344
        0           0     0.0313
```

Exercise 2.4.

If \mathbf{I} is the identity matrix in $\mathbb{R}^{n \times n}$, show that

$$\begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-\ell} & \dots & (-2)^{n-\ell} \\ & & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & & 0 & 1 & -2 \\ 0 & & & \dots & 0 & 0 & 1 \end{bmatrix} = \mathbf{I}.$$

2.2 Block Multiplication

Review: A rectangular matrix \mathbf{A} can be partitioned into submatrices by drawing horizontal lines between selected rows and vertical lines between selected columns. For example, the matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

can be partitioned as

$$(i) \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} = \left[\begin{array}{c|cc} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right], \quad (ii) [\mathbf{a}_{\cdot 1}, \mathbf{a}_{\cdot 2}, \mathbf{a}_{\cdot 3}] = \left[\begin{array}{ccc|c} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right],$$

$$(iii) \begin{bmatrix} \mathbf{a}_{\cdot 1}^T \\ \mathbf{a}_{\cdot 2}^T \\ \mathbf{a}_{\cdot 3}^T \end{bmatrix} = \left[\begin{array}{ccc|c} 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right], \quad (iv) [\mathbf{A}_{11}, \mathbf{A}_{12}] = \left[\begin{array}{cc|cc} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{array} \right].$$

In (i) the matrix \mathbf{A} is divided into four submatrices

$$\mathbf{A}_{11} = [1], \mathbf{A}_{12} = [2, 3], \mathbf{A}_{21} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}, \text{ and } \mathbf{A}_{22} = \begin{bmatrix} 5 & 6 \\ 8 & 9 \end{bmatrix},$$

while in (ii) and (iii) \mathbf{A} has been partitioned into columns and rows, respectively. The submatrices in a partition is often referred to as **blocks** and a partitioned matrix is sometimes called a **block matrix**. *

Exercise 2.5. Block Multiplication

1. Let $\mathbf{A}_{11} = \begin{bmatrix} 1 & 2 & -3 \\ 2 & 0 & -1 \end{bmatrix}$, $\mathbf{A}_{12} = \begin{bmatrix} -3 & 1 \\ -1 & 2 \end{bmatrix}$, $\mathbf{A}_{21} = \begin{bmatrix} -1 & 1 & -1 \end{bmatrix}$, $\mathbf{A}_{22} = \begin{bmatrix} -1 & 1 \end{bmatrix}$, $\mathbf{B}_{11} = \begin{bmatrix} 2 & 3 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}$, $\mathbf{B}_{12} = \begin{bmatrix} 2 & -3 \\ 2 & 0 \\ 0 & -1 \end{bmatrix}$, $\mathbf{B}_{21} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix}$, $\mathbf{B}_{22} = \begin{bmatrix} -2 & -3 \\ 0 & -1 \end{bmatrix}$. Compute

$$\begin{aligned} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21}, \quad \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22}, \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21}, \quad \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22}. \end{aligned}$$

2. We define $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix}$.

Compute \mathbf{AB} and compare with $\begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}$.

3. With MATLAB, create a file `mat2.m` where the matrices A_{ij} , B_{ij} are formed, then A and B . Compute the previous products.
4. Back to the general case. Show that if $B = [b_{.1}, \dots, b_{.n}]$ is partitioned by columns then the partition of the product AB by columns is

$$AB = [Ab_{.1}, Ab_{.2}, \dots, Ab_{.n}].$$

5. If $A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$ give conditions on the dimensions to get

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \times \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} = \begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{bmatrix}.$$

6. Suppose $A, B, C \in \mathbb{R}^{n \times n}$ are given in block form by

$$A := \begin{bmatrix} \lambda & \mathbf{a}^T \\ \mathbf{0} & A_1 \end{bmatrix}, \quad B := \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & B_1 \end{bmatrix}, \quad C := \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & C_1 \end{bmatrix},$$

where $A_1, B_1, C_1 \in \mathbb{R}^{(n-1) \times (n-1)}$. Show that

$$CAB = \begin{bmatrix} \lambda & \mathbf{a}^T B_1 \\ \mathbf{0} & C_1 A_1 B_1 \end{bmatrix}$$

Exercise 2.6.

Show that under suitable conditions the inverse of the block upper triangular matrix $A = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix}$ is $A^{-1} = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22}^{-1} \\ 0 & A_{22}^{-1} \end{bmatrix}$.

Exercise 2.7.

If $A \in \mathbb{R}^{n \times n}$ is nonsingular and upper triangular, show that A^{-1} is also upper triangular. ▷ *Math Hint*¹ ◁

2.3 Diagonally Dominant Matrices

◀ **Review:** A matrix $A \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$) is said to be **strictly diagonally dominant** if $|a_{ii}| > \sum_{j \neq i} |a_{ij}|$ for $i = 1, \dots, n$. We will show the following theorem.

Theorem 2.1. Suppose A is strictly diagonally dominant. Then the linear system $Ax = b$ has a unique solution for any $b \in \mathbb{R}^n$ (or \mathbb{C}^n). Moreover the solution x of $Ax = b$ is bounded as follows:

¹ Use induction and Exercise 2.6 with A_{22} a 1×1 matrix.

$$\max_{1 \leq j \leq n} |x_j| \leq \max_{1 \leq j \leq n} \frac{|b_j|}{\sigma_j}, \quad (2.1)$$

where $\sigma_i := |a_{ii}| - \sum_{j \neq i} |a_{ij}|$ for $i = 1, \dots, n$.

¶

Exercise 2.8.

1. Show the bound (2.1). ▷ Math Hint² ◷
2. Using (2.1) show that A is nonsingular.
3. Using MATLAB write a function `ub=ddboun(A, b)` that for given $A \in \mathbb{R}^{n \times n}$ and $b \in \mathbb{R}^n$ computes the upper bound ub in (2.1) if A is strictly diagonally dominant and set $ub = -1$ otherwise. Test on

$$A = \begin{bmatrix} 4 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 4 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}.$$

```
>> ub = ddboun(A, b)
ub =
    0.5000
```

4. Write a program `mat3.m` that constructs the triadiagonal matrix

$$A = \begin{bmatrix} -3 & 1 & 0 & \dots \\ 1 & -3 & 1 & 0 & \dots \\ 0 & 1 & -3 & 1 & 0 & \dots \\ & \ddots & \ddots & \ddots & \ddots \\ \dots & 0 & 1 & -3 & 1 \\ & \dots & 0 & 1 & -3 \end{bmatrix} \in \mathbb{R}^{n \times n} \text{ for a given } n. \text{ Then use}$$

`ddboun` for $b = [1, 1, \dots, 1]^T$. Test $n = 4$ then $n = 10$. ▷ MATLAB Hint³ ◷

```
n =
4
A =
-3      1      0      0
 1     -3      1      0
 0      1     -3      1
 0      0      1     -3
ub =
 1
```

² Let x be any solution of $Ax = b$. Choose an i so that $|x_i| = \max_j |x_j|$. Show that $|b_i| \geq |x_i| \sigma_i$.

³ Use the MATLAB functions `diag` and `ones`.

2.4 LU-Factorization of Diagonally Dominant Tridiagonal Systems

Review: Given a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} = \text{tridiag}(a_i, d_i, c_i) \in \mathbb{R}^{n \times n}$ is strictly diagonally dominant, tridiagonal, but not necessarily symmetric, we want to construct an LU-factorization $\mathbf{A} = \mathbf{L}\mathbf{U}$ of the form

$$\begin{bmatrix} d_1 & c_1 & & & \\ a_2 & d_2 & c_2 & & \\ \ddots & \ddots & \ddots & \ddots & \\ & a_{n-1} & d_{n-1} & c_{n-1} & \\ & & a_n & d_n & \end{bmatrix} = \begin{bmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & \ddots & \ddots & & \\ & & l_n & 1 & \end{bmatrix} \begin{bmatrix} u_1 & c_1 & & & \\ & \ddots & \ddots & & \\ & & u_{n-1} & c_{n-1} & \\ & & & u_n & \end{bmatrix}. \quad (2.2)$$

Note that \mathbf{L} has ones on the diagonal, and that \mathbf{A} and \mathbf{U} have the same superdiagonal. ■

Exercise 2.9.

1. Show by equating entries in (2.2) that

$$u_1 = d_1, \quad l_k = \frac{a_k}{u_{k-1}}, \quad u_k = d_k - l_k c_{k-1}, \quad k = 2, 3, \dots, n. \quad (2.3)$$

2. Show that the solution of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$ is found by solving $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$ as follows:

$$\begin{aligned} y_1 &= b_1, & y_k &= b_k - l_k y_{k-1}, & k &= 2, 3, \dots, n, \\ x_n &= y_n/u_n, & x_k &= (y_k - c_k x_{k+1})/u_k, & k &= n-1, \dots, 2, 1. \end{aligned} \quad (2.4)$$

Show that the process just described is well defined using the following steps.

3. Show that $|u_k| > |c_k|$ for $k = 1, 2, \dots, n-1$ and that $|u_n| > 0$. ▷ *Math Hint*⁴ ◁
4. Show that the LU-factorization exists and is unique.
5. Write a function `mylu.m` with the vectors $\mathbf{a}, \mathbf{d}, \mathbf{c}$ as input and the vectors $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{l} \in \mathbb{R}^{n-1}$ for output. The dimension n is obtained by the dimension of \mathbf{d} (see function `length`). All the vectors can be defined in \mathbb{R}^n provided that $a(1) = l(1) = 0, c(n) = 0$.

```
>> [l, u] = mylu([0 1 1 1 1], [4 4 4 4 4], [1 1 1 1 0])
l =
    0     0.2500    0.2667    0.2679    0.2679
u =
    4.0000    3.7500    3.7333    3.7321    3.7321
```

Even if the matrix is not strictly diagonal dominant, you can test

```
[l, u] = mylu([0 1 1 1 1]', [2 2 2 2 2], [1 1 1 1 0]).
```

⁴ Use induction on k .

2.5 Symmetric Positive Definite Systems and Cholesky Factorization

Review: A matrix $A \in \mathbb{R}^{n \times n}$ is said to be **symmetric** if $A^T = A$ and **positive definite** if $x^T A x > 0$ for all nonzero $x \in \mathbb{R}^n$. A is symmetric and positive definite if and only if it has a **Cholesky factorization** $A = LL^T$, where $L \in \mathbb{R}^{n \times n}$ is lower triangular with positive diagonal elements. The Cholesky factorization is unique if it exists. ♣

Baron André-Louis Cholesky (1875–1918) was a French military officer and mathematician. He worked in geodesy and map-making and he is known for the matrix decomposition (now Cholesky decomposition). As engineer officer, he was killed in battle a few months before the end of World War I.



Exercise 2.10.

Given the matrix $A := \begin{bmatrix} 4 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 \\ 2 & 0 & 5 & 0 \\ 0 & 2 & 0 & 5 \end{bmatrix}$.

1. Show that A is symmetric and positive definite.
2. Find the Cholesky factorization $A = LL^T$ of A .
3. Show that the Cholesky factorization of $B := \begin{bmatrix} 4 & 4 & 2 & 0 \\ 4 & 3 & 0 & 2 \\ 2 & 0 & 5 & 0 \\ 0 & 2 & 0 & 5 \end{bmatrix}$ does not exist and conclude that B is not positive definite.
4. With MATLAB, use the library function `chol` to test the factorization of the previous matrices A and B . ◁ MATLAB Hint⁵ ▷

2.6 Solutions

2.6.1 Some Matrix Computations

Exercise 2.1

```
>> A=[4,1,0,0;1,4,1,0;0,1,4,1;0,0,1,4]
A =
    4     1     0     0
    1     4     1     0
    0     1     4     1
    0     0     1     4
```

⁵ Note that `chol` produces an upper triangular matrix R so that $A = R^T R$.

```
|>> det(A)
ans =
209|
```

Exercise 2.2

$$\begin{aligned}\det A &= \begin{vmatrix} 2 & 1 & 3 & 3 \\ 1 & 0 & -1 & 0 \\ a & 2 & 1 & 1 \\ 4 & 3 & 2 & 4 \end{vmatrix} = \begin{vmatrix} 2 & 1 & 3 & 3 \\ 0 & -1/2 & -5/2 & -3/2 \\ 0 & 2-a/2 & 1-3a/2 & 1-3a/2 \\ 0 & 1 & -4 & -2 \end{vmatrix} \\ &= \begin{vmatrix} 2 & 1 & 3 & 3 \\ 0 & -1/2 & -5/2 & -3/2 \\ 0 & 0 & -9+a & -5 \\ 0 & 0 & -9 & -5 \end{vmatrix} \\ &= - \begin{vmatrix} 2 & 1 & 3 & 3 \\ 0 & -1/2 & -5/2 & -3/2 \\ 0 & 0 & -9 & -5 \\ 0 & 0 & -9+a & -5 \end{vmatrix} \\ &= - \begin{vmatrix} 2 & 1 & 3 & 3 \\ 0 & -1/2 & -5/2 & -3/2 \\ 0 & 0 & -9 & -5 \\ 0 & 0 & 0 & -5a/9 \end{vmatrix} = 5a\end{aligned}$$

```
|>> a=1;A=[2 1 3 3;1 0 -1 0;a 2 1 1;4 3 2 4];det(A)
ans =
5
>> rank(A)
ans =
4
>> a=0;A=[2 1 3 3;1 0 -1 0;a 2 1 1;4 3 2 4];det(A)
ans =
0
>> rank(A)
ans =
3|
```

Exercise 2.3

1.

$$B = \begin{bmatrix} 0 & 4 & 6 \\ 0 & 0 & 3 \\ 0 & 0 & 0 \end{bmatrix}, \quad B^2 = \begin{bmatrix} 0 & 0 & 12 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad B^n = \mathbf{0}, \quad n \geq 3.$$

2. Since $\mathbf{B}^n = \mathbf{0}$ for $n \geq 3$,

$$\begin{aligned}\mathbf{A}^n &= (2\mathbf{I} + \mathbf{B})^n = 2^n(\mathbf{I} + \mathbf{B}/2)^n = 2^n\left(\mathbf{I} + \frac{n}{2}\mathbf{B} + \frac{n(n-1)}{8}\mathbf{B}^2\right) \\ &= 2^{n-1} \begin{bmatrix} 2 & 4n & 3n(n+1) \\ 0 & 2 & 3n \\ 0 & 0 & 2 \end{bmatrix}\end{aligned}$$

Comment: The binomial expansion for matrices $(\mathbf{A} + \mathbf{B})^n = \sum_{k=0}^n \binom{n}{k} \mathbf{A}^k \mathbf{B}^{n-k}$ is valid if $\mathbf{AB} = \mathbf{BA}$. It does not hold in general. ☺

3. $(\mathbf{I} + \frac{\mathbf{B}}{2}) \times (\mathbf{I} - \frac{\mathbf{B}}{2} + \frac{\mathbf{B}^2}{4}) = \mathbf{I} - \frac{\mathbf{B}^3}{8} = \mathbf{I}$. Thus, $\mathbf{AC} = \mathbf{I}$, where $\mathbf{C} = \frac{1}{2}(\mathbf{I} - \frac{\mathbf{B}}{2} + \frac{\mathbf{B}^2}{4}) = \mathbf{A}^{-1}$. We find

$$\mathbf{A}^{-1} = \begin{bmatrix} \frac{1}{2} & -1 & 0 \\ 0 & \frac{1}{2} & -\frac{3}{4} \\ 0 & 0 & \frac{1}{2} \end{bmatrix}$$

4. We have

$$\mathbf{A}^{-1} = \frac{1}{2}(\mathbf{I} - \mathbf{E}), \quad \mathbf{E} = \begin{bmatrix} 0 & 2 & 0 \\ 0 & 0 & \frac{3}{2} \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{E}^2 = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{E}^n = \mathbf{0}, \quad n \geq 3.$$

Using the binomial expansion as above gives

$$\mathbf{A}^{-n} = 2^{-n-1} \begin{bmatrix} 2 & -4n & 3(n-1)n \\ 0 & 2 & -3n \\ 0 & 0 & 2 \end{bmatrix}$$

5. We have $\mathbf{A}^x \mathbf{A}^y = F(x)F(y) = F(x+y) = \mathbf{A}^{x+y}$ and $(\mathbf{A}^x)^y = F(x)^y = F(xy) = \mathbf{A}^{xy}$.

6. The file `mat1.m`

```
n=5
A=[2 ,4 ,6;0 ,2 ,3;0 ,0 ,2];
disp([ 'A^( ',int2str(n),') : '])
A^n
disp([ 'A^( ',int2str(-n),') : '])
inv(A)^n
```

Exercise 2.4

$$\text{Let } \mathbf{A} = \begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-\ell} & \dots & (-2)^{n-\ell} \\ & & & & \ddots & \ddots & \vdots \\ 0 & & & & & 0 & 1 & -2 \\ 0 & & \dots & & & 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{C} = \mathbf{A} \times \mathbf{B}$.

$$\text{We notice that } a_{i,\ell} = \begin{cases} 1 & \text{if } \ell = i \\ 2 & \text{if } \ell = i+1 \\ 0 & \text{if } \ell \notin \{i, i+1\} \end{cases} \text{ and } b_{\ell,j} = \begin{cases} (-2)^{j-\ell} & \text{if } \ell < j \\ 1 & \text{if } \ell = j \\ 0 & \text{if } \ell > j \end{cases}$$

We deduce that

$$c_{i,j} = \sum_{\ell=1}^n a_{i,\ell} b_{\ell,j} = \begin{cases} 1 \times (-2)^{j-i} + 2 \times (-2)^{j-(i+1)} = 0 & \text{for } j > i \\ 1 \times (-2)^{j-i} = 1 & \text{for } j = i \\ 1 \times 0 + 2 \times 0 = 0 & \text{for } j < i \end{cases}$$

2.6.2 Block Multiplication**Exercise 2.5**

1.

$$\mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} = \begin{bmatrix} 4 & 3 \\ 8 & 10 \end{bmatrix}, \quad \begin{bmatrix} 12 & 8 \\ 6 & -4 \end{bmatrix} = \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22},$$

$$\mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} = \begin{bmatrix} 0 & -2 \end{bmatrix}, \quad \begin{bmatrix} 2 & 6 \end{bmatrix} = \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22}.$$

2.

$$\mathbf{AB} = \begin{bmatrix} 4 & 3 & 12 & 8 \\ 8 & 10 & 6 & -4 \\ 0 & -2 & 2 & 6 \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}.$$

3. mat2.m

```
A11=[1,2,-3;2,0,-1];
A12=[-3,1;-1,2];
A21=[-1,1,-1];
A22=[-1,1];
A=[A11,A12;A21,A22];

B11=[2 3;0 -1;-1 -1];
B12=[2 -3;2 0;0 -1];
B21=[1 1;2 2];
```

```

B22=[-2 -3;0 -1];
B=[B11 ,B12 ;B21 ,B22 ];

disp(' Block multiplication ')
C11=A11*B11+A12*B21
C12=A11*B12+A12*B22
C21=A21*B11+A22*B21
C22=A21*B12+A22*B22
C=[A11*B11+A12*B21 ,A11*B12+A12*B22 ;...
    A21*B11+A22*B21 ,A21*B12+A22*B22 ]
disp(' Direct computation ')
D=A*B

```

4. Let $\mathbf{A} \in \mathbb{R}^{m \times p}$ and $\mathbf{B} \in \mathbb{R}^{p \times n}$. If $\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m \times n}$ then

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj} = [a_{i1} \ a_{i2} \ \dots \ a_{ip}] \times \begin{bmatrix} b_{1j} \\ b_{2j} \\ \vdots \\ b_{nj} \end{bmatrix} = [a_{i1} \ a_{i2} \ \dots \ a_{ip}] \times \mathbf{b}_{\cdot j}.$$

Hence $\mathbf{c}_{\cdot j} = \mathbf{Ab}_{\cdot j}$.

5. For $i, j, k = 1, 2$, we suppose that $\mathbf{A}_{ik} \in \mathbb{R}^{m_{ik} \times n_{ik}}$ and $\mathbf{B}_{kj} \in \mathbb{R}^{p_{kj} \times q_{kj}}$. The computations $\mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21}$, $\mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22}$, $\mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21}$, $\mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22}$ are possible provided that $n_{11} = p_{11} = n_{21} = p_{12}$ and $n_{12} = p_{21} = n_{22} = p_{22}$.

Conversely, if the dimensions satisfy the previous equalities, then

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} \\ \mathbf{B}_{21} & \mathbf{B}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11}\mathbf{B}_{11} + \mathbf{A}_{12}\mathbf{B}_{21} & \mathbf{A}_{11}\mathbf{B}_{12} + \mathbf{A}_{12}\mathbf{B}_{22} \\ \mathbf{A}_{21}\mathbf{B}_{11} + \mathbf{A}_{22}\mathbf{B}_{21} & \mathbf{A}_{21}\mathbf{B}_{12} + \mathbf{A}_{22}\mathbf{B}_{22} \end{bmatrix}.$$

6. If $\mathbf{A} := \begin{bmatrix} \lambda & \mathbf{a}^T \\ \mathbf{0} & \mathbf{A}_1 \end{bmatrix}$, $\mathbf{B} := \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{B}_1 \end{bmatrix}$ and $\mathbf{C} := \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{C}_1 \end{bmatrix}$, with $\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1 \in \mathbb{R}^{(n-1) \times (n-1)}$, then

$$\begin{aligned} \mathbf{CAB} &= \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{C}_1 \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{a}^T \\ \mathbf{0} & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{B}_1 \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{0} & \mathbf{C}_1 \end{bmatrix} \begin{bmatrix} \lambda & \mathbf{a}^T \mathbf{B}_1 \\ \mathbf{0} & \mathbf{A}_1 \mathbf{B}_1 \end{bmatrix} \\ &= \begin{bmatrix} \lambda & \mathbf{a}^T \mathbf{B}_1 \\ \mathbf{0} & \mathbf{C}_1 \mathbf{A}_1 \mathbf{B}_1 \end{bmatrix}. \end{aligned}$$

Exercise 2.6

Let $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \mathbf{A}_{22} \end{bmatrix} \in \mathbb{R}^{n \times n}$. If \mathbf{A}_{11} and \mathbf{A}_{22} have an inverse then those matrices are square and $\mathbf{A}_{11} \in \mathbb{R}^{p \times p}$ and $\mathbf{A}_{22} \in \mathbb{R}^{q \times q}$ with $p + q = n$. Moreover let $\mathbf{B} = \begin{bmatrix} \mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1} \mathbf{A}_{12} \mathbf{A}_{22}^{-1} \\ 0 & \mathbf{A}_{22}^{-1} \end{bmatrix}$. Then with the results of the previous Exercise,

$\mathbf{A} \times \mathbf{B} = \mathbf{B} \times \mathbf{A} = \mathbf{I}_{n \times n}$ so that $\mathbf{B} = \mathbf{A}^{-1}$.

Exercise 2.7

Let us suppose that $\mathbf{A} \in \mathbb{R}^{n \times n}$ is nonsingular and upper triangular.

If $n = 1$, then $\mathbf{A} = [a_{11}] \neq 0$ and $\mathbf{A}^{-1} = [1/a_{11}]$ is upper triangular.

If $n > 1$, let us write $\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ 0 & \mathbf{A}_{22} \end{bmatrix}$, where $\mathbf{A}_{11} \in \mathbb{R}^{(n-1) \times (n-1)}$ is nonsingular and upper triangular. Assume that the matrix \mathbf{A}_{11}^{-1} is upper triangular. Then by Exercise 2.6 the matrix \mathbf{A}^{-1} is upper triangular.

2.6.3 Diagonally Dominant Matrices**Exercise 2.8**

1.

$$|b_i| = |a_{ii}x_i + \sum_{j \neq i} a_{ij}x_j| \geq |x_i|(|a_{ii}| - \sum_{j \neq i} |a_{ij}|) = |x_i|\sigma_i.$$

Thus

$$\max_{1 \leq j \leq n} |x_j| = |x_i| \leq \frac{|b_i|}{\sigma_i} \leq \max_{1 \leq j \leq n} \left(\frac{|b_j|}{\sigma_j} \right).$$

2. If $\mathbf{Ax} = \mathbf{0}$ then $\max_{1 \leq j \leq n} |x_j| \leq 0$ by (2.1), and so $\mathbf{x} = \mathbf{0}$.

3. `ddbound.m`

```
function ub=ddbound(A, b)
B=abs(A);
s=2*diag(B)-sum(B, 2);
if s>0
    ub=max(b ./ s);
else
    ub=-1;
end
```

4. `mat3.m`

```
n=4
A=-3*diag(ones(n,1))+diag(ones(n-1,1),1)...
+diag(ones(n-1,1),-1)
b=ones(n,1);
ub=ddbound(A, b)
```

2.6.4 LU-Factorization of Diagonally Dominant Tridiagonal Systems**Exercise 2.9**

- The product of \mathbf{L} and \mathbf{U} is a tridiagonal matrix that must be equal to \mathbf{A} . Comparing elements in row 1 gives $[u_1, c_1] = [d_1, c_1]$ showing that $u_1 = d_1$. For row

k we get the equations $[l_k u_{k-1}, l_k c_{k-1} + u_k, c_k] = [a_k, d_k, c_k]$ for $k = 2, \dots, n$, and solving for l_k and u_k gives (2.3).

$$2. \text{ Let } \mathbf{Ax} = \mathbf{b} \text{ and } \mathbf{y} := \mathbf{Ux}. \text{ Then } \mathbf{Ly} = \mathbf{b} \text{ or } \begin{bmatrix} 1 & & & & & \\ l_2 & 1 & & & & \\ \ddots & \ddots & \ddots & & & \\ & & & l_n & 1 & \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} =$$

$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$. Solving the first equation we find $y_1 = b_1$ and solving the k th equation

$l_k y_{k-1} + y_k = b_k$ for $y_k, k = 2, \dots, n$ we obtain the first part of (2.4). The

system for \mathbf{x} is upper triangular $\begin{bmatrix} u_1 & c_1 & & & & \\ \ddots & \ddots & \ddots & & & \\ & u_{n-1} & c_{n-1} & & & \\ & & u_n & & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ y_{n-1} \\ y_n \end{bmatrix}$.

Solving the last equation for x_n and then the k th equation for x_k in the order $n-1, n-2, \dots, 1$ show the second part of (2.4).

3. By strict diagonal dominance $|u_1| = |d_1| > |c_1|$. Suppose $|u_{k-1}| > |c_{k-1}|$ for some $k < n$. By (2.3) and strict diagonal dominance

$$|u_k| = |d_k - \frac{a_k}{u_{k-1}} c_{k-1}| \geq |d_k| - |a_k| \frac{|c_{k-1}|}{|u_{k-1}|} \geq |d_k| - |a_k| > |c_k|.$$

This also holds for $k = n$ if we define $c_n = 0$. Thus the claim follows by induction.

4. We have shown that $u_k \neq 0$ for $k = 1, \dots, n$. But then the formulas (2.3) are well defined giving an LU-factorization of \mathbf{A} . Since any LU factorization of \mathbf{A} must satisfy these equations, uniqueness follows.
5. mylu.m

```
function [l, u]=mylu(a, d, c)
n=length(d);
u(1)=d(1);
for k=2:n
    l(k)=a(k)/u(k-1);
    u(k)=d(k)-l(k)*c(k-1);
end
```

2.6.5 Symmetric Positive Definite Systems and Cholesky Factorization

Exercise 2.10

1. Since $a_{ij} = a_{ji}$ for all i, j , the matrix is symmetric. We find

$$\begin{aligned}\mathbf{x}^T \mathbf{A} \mathbf{x} &= [x_1, x_2, x_3, x_4] \begin{bmatrix} 4 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 \\ 2 & 0 & 5 & 0 \\ 0 & 2 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} \\ &= 4x_1^2 + 2x_1x_3 + 4x_2^2 + 2x_2x_4 + 2x_3x_1 + 5x_3^2 + 2x_4x_2 + 5x_4^2 \\ &= (2x_1 + x_3)^2 + 4x_3^2 + (2x_2 + x_4)^2 + 4x_4^2.\end{aligned}$$

Thus $\mathbf{x}^T \mathbf{A} \mathbf{x} \geq 0$ and if $\mathbf{x}^T \mathbf{A} \mathbf{x} = 0$ then $2x_1 + x_3 = 0$, $x_3 = 0$, $2x_2 + x_4 = 0$, and $x_4 = 0$. This implies that $\mathbf{x} = \mathbf{0}$ and the matrix is symmetric and positive definite.

2. We determine $\mathbf{L} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix}$ so that $\mathbf{A} = \mathbf{L} \mathbf{L}^T$. This can be done

either column by column or row by row. Consider the column approach. The first column of $\mathbf{L} \mathbf{L}^T$ is equal to the first column of \mathbf{A} , and therefore

$$[l_{11}^2, l_{21}l_{11}, l_{31}l_{11}, l_{41}l_{11}]^T = [a_{11}, a_{21}, a_{31}, a_{41}]^T. \quad (2.5)$$

Since $a_{11} = 4 > 0$, we can take a real square root and the first column of \mathbf{L} is given by $l_{11} = 2$. But then $l_{21} = 0$, $l_{31} = 1$, and $l_{41} = 0$. Using this information we can determine the second column of \mathbf{L} by equating the lower part of the second column of $\mathbf{L} \mathbf{L}^T$ with the corresponding part of the second column of \mathbf{A} . Thus

$$\begin{bmatrix} l_{21}^2 + l_{22}^2 \\ l_{31}l_{21} + l_{32}l_{22} \\ l_{41}l_{21} + l_{42}l_{22} \end{bmatrix} = \begin{bmatrix} a_{22} \\ a_{32} \\ a_{42} \end{bmatrix}, \quad (2.6)$$

and we obtain $l_{22} = 2$, $l_{32} = 0$, $l_{42} = 1$. For the third column the last two lines give

$$\begin{bmatrix} l_{31}^2 + l_{32}^2 + l_{33}^2 \\ l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} \end{bmatrix} = \begin{bmatrix} 1 + l_{33}^2 \\ l_{43}l_{33} \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \text{ with solution } l_{33} = 2 \text{ and}$$

$l_{43} = 0$. Finally for the last line the element l_{44} is determined from $l_{41}^2 + l_{42}^2 +$

$$l_{43}^2 + l_{44}^2 = 0^2 + 1^2 + 0^2 + l_{44}^2 = 5 \text{ giving } l_{44} = 2. \text{ Thus } \mathbf{L} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}.$$

3. We try to find the Cholesky factorization of \mathbf{B} and proceed as above. From (2.5) we find $[l_{11}^2, l_{21}l_{11}, l_{31}l_{11}, l_{41}l_{11}]^T = [4, 4, 2, 0]^T$ giving $l_{11} = 2$, $l_{21} = 2$, $l_{31} = 1$, and $l_{41} = 0$. For the second column (2.5) gives the equations

$$\begin{bmatrix} l_{21}^2 + l_{22}^2 \\ l_{31}l_{21} + l_{32}l_{22} \\ l_{41}l_{21} + l_{42}l_{22} \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix}.$$

But then l_{22} is the solution of the equation $4 + l_{22}^2 = 3$ and this equation does not have a real solution. Thus the Cholesky factorization of \mathbf{B} does not exist. We conclude that \mathbf{B} is not positive definite since a matrix has a Cholesky factorization if and only if it is symmetric and positive definite.

4. test chol

```
>> A=[4,0,2,0;0,4,0,2;2,0,5,0;0,2,0,5]
A =
    4      0      2      0
    0      4      0      2
    2      0      5      0
    0      2      0      5
>> chol(A)
ans =
    2      0      1      0
    0      2      0      1
    0      0      2      0
    0      0      0      2
>> B=[4,4,2,0;4,3,0,2;2,0,5,0; 0,2,0,5]
B =
    4      4      2      0
    4      3      0      2
    2      0      5      0
    0      2      0      5
>> chol(B)
Error using chol
Matrix must be positive definite.
```

Chapter 3

Matrices, Eigenvalues and Eigenvectors

Many complex mathematical problems can be formulated as eigenvalue problems. In Exercises 3.1 and 3.2 examples are given where direct computation of eigenvalues and eigenspaces can be carried out. In Exercise 3.3 we show how the eigenvalues of a matrix and its inverse are related, while the eigenvalues of a positive definite matrix are considered in Exercise 3.4. In Exercise 3.5 we see how limits of powers of matrices can be computed via eigenvalues and eigenvectors.

In Sect. 3.2 we study how a single eigenvalue and eigenvector can be computed using powers of matrices. In Exercise 3.6 the dominant eigenvalue is computed, while in Exercise 3.7 we demonstrate on an example how powers of shifted inverses can be used to find an eigenvalue. This method, also known as the **Rayleigh quotient iteration** converges quadratically and even cubically for symmetric matrices.

To find all eigenvalues and eigenvectors of a matrix the **QR-method** is the standard choice. We consider this method in two Exercises 3.8 and 3.9.

Finally, an application of eigenvalues is considered in Sect. 3.4. We determine all eigenvalues and eigenvectors of a special test matrix in Exercise 3.10 and use this in Exercise 3.11 to determine whether a beam will break when it is subject to a longitudinal force.

Review: We say that $\lambda \in \mathbb{C}$ is an **eigenvalue** of $A \in \mathbb{C}^{n \times n}$ if $Ax = \lambda x$ for a nonzero $x \in \mathbb{C}^n$. The vector x is called an **eigenvector**. The set of corresponding eigenvectors is the subspace $\ker(A - \lambda I)$ called an **eigenspace**.

The eigenvalues $\lambda_1, \dots, \lambda_n$ are the roots of the **characteristic polynomial**.

$$\pi_A(t) := \det(A - tI) = \begin{vmatrix} a_{11} - t & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - t & \cdots & a_{2n} \\ \ddots & & & \\ a_{n1} & a_{n2} & \cdots & a_{nn} - t \end{vmatrix} = (\lambda_1 - t) \cdots (\lambda_n - t).$$

The **multiplicity** of an eigenvalue is its multiplicity as a root in π_A , which is at least the dimension of the corresponding eigenspace. The set $\sigma(A) := \{\lambda_1, \dots, \lambda_n\}$ of eigenvalues of A is called the **spectrum** of A . Eigenvalues are related to the **trace**

and the **determinant** as follows: $\text{trace}(\mathbf{A}) := \sum_{i=1}^n a_{ii} = \sum_{i=1}^n \lambda_i$, $\det(\mathbf{A}) = \prod_{i=1}^n \lambda_i$.

The **spectral radius** of \mathbf{A} is given by $\rho(\mathbf{A}) := \max\{|\lambda| : \lambda \in \sigma(\mathbf{A})\}$. A localization of the eigenvalues can be obtained with the **Gershgorin circle Theorem** (8.2 with proof): If λ is an eigenvalue of $\mathbf{A} \in \mathbb{C}^{n \times n}$ then

$$\lambda \in \bigcup_{i=1}^n \mathcal{D}_i, \text{ where } \mathcal{D}_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j=1, j \neq i}^n |a_{ij}|\}.$$

The matrices $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times n}$ are **similar** if $\mathbf{B} = \mathbf{X}^{-1} \mathbf{A} \mathbf{X}$ for a nonsingular matrix $\mathbf{X} \in \mathbb{C}^{n \times n}$. Similar matrices have the same characteristic polynomial and therefore the same eigenvalues with the same multiplicities. If \mathbf{u} is an eigenvector of \mathbf{A} , then $\mathbf{X}^{-1} \mathbf{u}$ is an eigenvector of \mathbf{B} for the same eigenvalue. If \mathbf{B} is a diagonal matrix then the diagonal elements $\lambda_1, \dots, \lambda_n$ of \mathbf{B} are the eigenvalues of \mathbf{A} and the columns $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of \mathbf{X} are eigenvectors of \mathbf{A} , $\mathbf{A} \mathbf{x}_j = \lambda_j \mathbf{x}_j$, $j = 1, \dots, n$.

Conversely, \mathbf{A} is similar to a diagonal matrix \mathbf{B} with diagonal $\lambda_1, \dots, \lambda_n$ if and only if there exist a basis of corresponding eigenvectors $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ of \mathbf{A} . Moreover if \mathbf{X} is the matrix with these eigenvectors as columns then $\mathbf{B} := \mathbf{X}^{-1} \mathbf{A} \mathbf{X}$.

For $\mathbf{U} \in \mathbb{C}^{n \times n}$, we define the **conjugate transpose** matrix \mathbf{U}^* by $[u_{ij}^* = \bar{u}_{ji}]_{i,j=1,\dots,n}$. If $\mathbf{X} = \mathbf{U}$ is **unitary**, i.e., $\mathbf{U}^* \mathbf{U} = \mathbf{I}$, then $\mathbf{U}^{-1} = \mathbf{U}^*$ and $\mathbf{U}^* \mathbf{A} \mathbf{U}$ is a **unitary similarity transformation**.

A matrix $\mathbf{A} \in \mathbb{C}^{n \times n}$ is **Hermitian** if $\mathbf{A}^* = \mathbf{A}$. All its eigenvalues are real and it can be diagonalized by a unitary matrix $\mathbf{U} \in \mathbb{C}^{n \times n}$ of n orthonormal eigenvectors. For the real case, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is **symmetric** i.e. $a_{ij} = a_{ji}$ and $\mathbf{U} \in \mathbb{R}^{n \times n}$. ♦

3.1 Elementary Computations

Exercise 3.1. Computations of Eigenvalues and Eigenvectors

- Find eigenvalues and eigenvectors and eigenspaces of the following matrices

$$\mathbf{A} := \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{I} := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{B} := \begin{bmatrix} 5 & 3 & -3 \\ 1 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}, \quad \mathbf{C} := \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & -1 \\ 1 & 2 & 0 \end{bmatrix}.$$

- ▷ *Math Hint*¹ ◁
- Let $\mathbf{M} := \begin{bmatrix} m & 1 & 1 \\ 1 & m & 1 \\ 1 & 1 & m \end{bmatrix}$. Find the rank of $\mathbf{M} - t\mathbf{I}$ when $m - t = 1$ and deduce the eigenvalues of \mathbf{M} . ▷ *Math Hint*² ◁

¹ Compute $\det(\mathbf{M} - t\mathbf{I})$.

² Use $\text{trace}(\mathbf{M})$.

3. Let $\mathbf{J} := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. Show that \mathbf{J} is not similar to a diagonal matrix. ▷ Math Hint³ ◷

4. For the **Jordan block** $\mathbf{J}_n(\lambda) := \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ & & \ddots & \ddots & \\ 0 & \cdots & 0 & \lambda & 1 \\ 0 & \cdots & \cdots & 0 & \lambda \end{bmatrix} \in \mathbb{C}^{n \times n}$, show that

the only eigenvalue is λ and the eigenspace is of dimension 1 with direction the unit vector $e_1 = [1, 0, \dots, 0]^T$. ▷ Math Hint⁴ ◷

Marie Ennemond **Camille Jordan** (1838–1922) was a French mathematician who is known for his work in group theory and Galois theory and for his “Cours d’analyse de l’École Polytechnique”. He was an engineer from École Polytechnique where he became Professor. He also was Professor at the Collège de France.



Exercise 3.2. Block upper triangular form

Find the eigenvalues of the matrix $\mathbf{M} := \begin{bmatrix} 2 & 1 & 3 & 4 & 5 \\ -1 & 2 & 1 & 2 & 3 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 0 & 0 & 3 & 2 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$.

Exercise 3.3. Eigenvalues of the inverse matrix

If $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a non singular matrix, show that the eigenvalues of \mathbf{A}^{-1} are the inverses of the eigenvalues of \mathbf{A} .

Exercise 3.4. Eigenvalues of a positive definite matrix

If $\mathbf{A} \in \mathbb{R}^{n \times n}$ is a symmetric, positive definite matrix (see Exercise 2.10), show that its eigenvalues are real and positive.

Exercise 3.5. Power of a matrix

Let $\mathbf{A} = \begin{bmatrix} 0 & 1/4 & 3/4 \\ 3/4 & 0 & 1/4 \\ 1/4 & 3/4 & 0 \end{bmatrix}$. Using MATLAB,

1. Find the eigenvalues and eigenvectors of \mathbf{A} .
2. Construct the matrix \mathbf{P} such that $\mathbf{P}^{-1}\mathbf{AP}$ is a diagonal matrix.
3. Find $\lim_{n \rightarrow \infty} \mathbf{A}^n$.

³ Find the dimension of the eigenspace.

⁴ Consider the i -th row in $\mathbf{J}\mathbf{u} = \lambda\mathbf{u}$.

3.2 The Power and Inverse Power Methods

Exercise 3.6. The power method

We assume that $A \in \mathbb{R}^{n \times n}$ can be diagonalized by a similarity transformation and that its eigenvalues satisfy $|\lambda_1| > |\lambda_2| \geq \dots \geq |\lambda_n|$. We note that the **dominant eigenvalue** λ_1 is real. If not, the complex conjugate $\bar{\lambda}_1$ is also an eigenvalue with $|\lambda_1| = |\bar{\lambda}_1|$. We choose the 2-norm on \mathbb{R}^n or \mathbb{C}^n denoted $\|\cdot\|$ i.e. $\|x\|^2 = \sum_{i=1}^n |x_i|^2$.

Let u_1, \dots, u_n be a basis of eigenvectors. We start with a vector x_0 which is not in the subspace spanned by u_2, \dots, u_n . With the help of the following algorithm we construct sequences (μ_i) , (x_i) , (q_i) :

$$\boxed{\begin{aligned} q_0 &= \frac{x_0}{\|x_0\|} \\ \text{For } i = 1, 2, \dots, \\ x_i &= Aq_{i-1}, \\ q_i &= \frac{x_i}{\|x_i\|}, \\ \mu_i &= q_i^T A q_i \end{aligned}}$$

Then it can be shown that (μ_i) converges to the dominant eigenvalue λ_1 .

1. Start with a random matrix A of dimension 10 and a random vector x_0 and program the algorithm using *maxiter* iterations. Save the program in `power1.m`. Test with $\text{maxiter} = 50$. Compute the “error” $\|A * q_{50} - \mu_{50} q_{50}\|$.
2. Calculate the dominant eigenvalue λ using the MATLAB functions `eig`, `abs`, `max`. Compute the error $\mu_{50} - \lambda$. Save the program in `power2.m`. Use the same test data as before.
3. Combine `power1.m` and `power2.m` and add a plot showing $\mu(i)$ as a function of i . Save the program in `power3.m`. Same test data as before. See Fig. 3.1.

Comment: Because random numbers can differ, the dominant eigenvalue and the figure might be different. ☺

4. Replace the random matrix by $A_1 := \begin{bmatrix} 1 & 1 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & -0.99 \end{bmatrix}$ then $A_2 = \begin{bmatrix} 1 & 1 & -2 \\ 0 & -1 & -1 \\ 0 & 0 & 0.1 \end{bmatrix}$ with $\text{maxiter} = 100$.

Comment: For A_1 , the sequence (μ_i) still seems to converge to λ_1 . This is not the case for A_2 . The hypotheses are not satisfied in the two cases. The first case indicates that the conditions for convergence of the sequence (μ_i) are only sufficient. ☺

Exercise 3.7. The shifted inverse power method

Applying the power method to the matrix $(A - sI)^{-1}$ is known as the **shifted inverse power method** also known as the **Rayleigh quotient iteration**. The real or complex scalar s is called a **shift**.

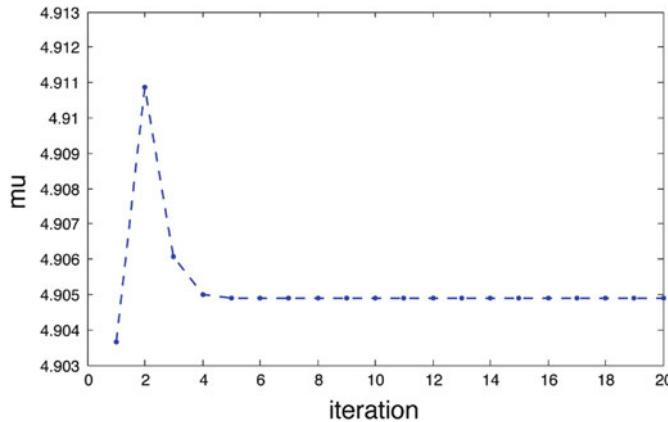


Fig. 3.1 Successive values of the approximations

John William Strutt, 3rd Baron Rayleigh (1842–1919) was an English physicist who discovered argon, earned the Nobel Prize for Physics in 1904. In mathematics, he introduced the now known Rayleigh quotient. Lord Rayleigh was elected Fellow of the Royal Society on 12 June 1873, and served as president of the Royal Society from 1905 to 1908.



The eigenvalues $\lambda_1, \dots, \lambda_n$ of \mathbf{A} are “shifted” to the eigenvalues $\mu_j := 1/(\lambda_j - s)$ of $(\mathbf{A} - s\mathbf{I})^{-1}$. Thus if λ_k is a simple eigenvalue then μ_k is a dominant eigenvalue if s is sufficiently close to λ_k . In the following algorithm we start with \mathbf{q}_0 with $\|\mathbf{q}_0\| = 1$ and a shift s_0 and change the shift in each iteration:

$$\begin{aligned} &\text{For } i = 1, 2, \dots, \\ &(\mathbf{A} - s_{i-1}\mathbf{I})\mathbf{x}_i = \mathbf{q}_{i-1}, \\ &\mathbf{q}_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}, \\ &s_i = \mathbf{q}_i^T \mathbf{A} \mathbf{q}_i \end{aligned}$$

1. Write a function `function [s1,q1]=invpower(A,s,q)` which given \mathbf{q} with $\|\mathbf{q}\| = 1$ and shift s calculates one iteration in the above algorithm returning new values for s_1 and \mathbf{q}_1 .

Example

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \quad s = 0, \quad \mathbf{q} = [1, 0, 0]^T.$$

If `invpower` is applied repeatedly starting with $s = 0$ it will converge to the smallest eigenvalue l of \mathbf{A} . Compute l using the MATLAB function `eig` and compute the error $l - s$ after each call to `invpower`.

```

>> A=[2,-1,0;-1,2,-1;0,-1,2];
>> mu=eig(A)
mu =
    0.5858
    2.0000
    3.4142
>> l=mu(1);
>> s=0; q=[1,0,0]';
>> [s,q]=invpower(A,s,q); 1-s
ans =
    -0.2714
>> [s,q]=invpower(A,s,q); 1-s
ans =
    -0.0145
>> [s,q]=invpower(A,s,q); 1-s
ans =
    -1.4871e-06
>> [s,q]=invpower(A,s,q); 1-s
ans =
    1.1102e-16

```

Also try starting with $s = 1.7$ and $\ell = \mu(2)$, then $\ell = 4$ and $\ell = \mu(3)\dots$

Comment: The convergence is very rapid. The error is approximately proportional to the third power of the previous error. This is known as **cubic convergence**. For a nonsymmetric matrix the convergence is **quadratic**. ☺

3.3 The QR Method

With the QR method one can compute all eigenvalues and eigenvectors of any matrix $A \in \mathbb{C}^{n \times n}$. We generate a sequence (A_k) of matrices that converges to an upper triangular matrix with the eigenvalues of A on the diagonal. If A is real with some complex eigenvalues and real arithmetic is used then (A_k) converges to a block upper triangular matrix. The diagonal blocks are of size 1 or 2 corresponding to real or complex eigenvalues of A , see Exercise 3.2 for an example of a block upper triangular matrix. Unitary similarity transformations are used in each iteration.

The decomposition $A = QR$ is used in the QR method. It is called a **QR factorization** if $Q \in \mathbb{C}^{n \times n}$ is unitary, i. e., $Q^*Q = I$ and $R \in \mathbb{C}^{n \times n}$ is upper triangular.

Exercise 3.8. QR factorization

- For a nonsingular $A \in \mathbb{R}^{n \times n}$ let $A^T A = R^T R$ be the Cholesky factorization of $A^T A$ (see Sect. 2.5 and Exercise 2.10), i. e., $R \in \mathbb{R}^{n \times n}$ is upper triangular with positive diagonal elements. Show that $Q := AR^{-1}$ is a unitary matrix. $A = QR$ is the **QR factorization** of A that has positive elements on the diagonal in R .
- Find in this way the QR factorization of $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.
- Show that

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 1 \\ -2 & -1 & 0 \\ 2 & 2 & 1 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} -1 & 2 & -2 \\ 2 & -1 & -2 \\ -2 & -2 & -1 \end{bmatrix} \begin{bmatrix} -3 & -2 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

is a QR factorization of \mathbf{A} . Verify using the MATLAB function `qr`.

Exercise 3.9. Basic QR algorithm

The **basic QR algorithm** takes the form:

```

 $\mathbf{A}_1 = \mathbf{A}$ 
for  $k = 1, 2, \dots$ 
     $\mathbf{A}_k = \mathbf{Q}_k \mathbf{R}_k$       (QR factorization of  $\mathbf{A}_k$ )
     $\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k$ .
end

```

Since $\mathbf{R}_k = \mathbf{Q}_k^* \mathbf{A}_k$ we see that $\mathbf{A}_{k+1} = \mathbf{Q}_k^* \mathbf{A}_k \mathbf{Q}_k$ so that \mathbf{A}_k and \mathbf{A}_{k+1} are unitary similar.

1. Watch the convergence of the basic QR algorithm by performing repeated QR step on the matrix $\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$.

Comment: Note that \mathbf{A} is symmetric. It appears that the sequence (\mathbf{A}_k) converges to a diagonal matrix with the eigenvalues on the diagonal. ☺

2. Repeat on $\mathbf{B} := \begin{bmatrix} 5 & 3 & -3 \\ 1 & 3 & 1 \\ 0 & 0 & 2 \end{bmatrix}$, $\mathbf{C} := \begin{bmatrix} 1 & 0 & 0 \\ 1 & 3 & -1 \\ 1 & 2 & 0 \end{bmatrix}$, whose eigenvalues have been computed in Exercise 3.1 and can also be checked via the MATLAB function `eig`.
3. Repeat also on $\mathbf{D} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Comment: In these cases, the sequence converges to an upper triangular matrix if the eigenvalues are real and a block upper triangular matrix if some of the eigenvalues are complex. ☺

3.4 Application to the Buckling of a Beam

Exercise 3.10. Derivative Matrix

In this exercise we determine for $m \geq 2$ the eigenvalues and eigenvectors of the tridiagonal matrix

$$\mathbf{T}_m := \begin{bmatrix} 2 & -1 & 0 & & & \\ -1 & 2 & -1 & & & \\ 0 & \ddots & \ddots & \ddots & & \\ & & & & 0 & \\ & & & & -1 & 2 & -1 \\ & & & & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{m \times m}. \quad (3.2)$$

These will then be used to analyze an eigenvalue problem for a two point boundary value problem and to give a fast method to solve the discrete Poisson problem on a square.

We set $h := 1/(m + 1)$.

1. Find the eigenvalues of \mathbf{T}_2 and show that they can be written $\lambda_j = 2(1 - \cos(j\pi h))$, $j = 1, 2$.
2. For $A, B \in \mathbb{R}$, show that $2 \sin A - \sin 2A = 2(1 - \cos(A)) \sin A$ and $-\sin(A - B) + 2 \sin A - \sin(A + B) = 2(1 - \cos(B)) \sin A$. \triangleright Math Hint⁵ \triangleleft
3. Show that $\mathbf{s}_j := [\sin(j\pi/3), \sin(2j\pi/3)]^T$ are eigenvectors of \mathbf{T}_2 for $j = 1, 2$, $\mathbf{s}_1^T \mathbf{s}_2 = 0$ and $\mathbf{s}_j^T \mathbf{s}_j = 1/(2h)$, $j = 1, 2$.
4. For $m \geq 2$ show that $\mathbf{T}_m \mathbf{s}_j = \lambda_j \mathbf{s}_j$, $j = 1, \dots, m$, where

$$\mathbf{s}_j = [\sin(j\pi h), \sin(2j\pi h), \dots, \sin(mj\pi h)]^T, \quad (3.3)$$

$$\lambda_j = 2(1 - \cos(j\pi h)). \quad (3.4)$$

5. Prove that $\mathbf{s}_j^T \mathbf{s}_\ell = 0$ for $j \neq \ell$.
6. Show that $\sum_{k=0}^m e^{2ikj\pi h} = 0$ for $j = 1, \dots, m$. \triangleright Math Hint⁶ \triangleleft
7. Show that $\mathbf{s}_j^T \mathbf{s}_j$ is independent of $j = 1, \dots, m$. \triangleright Math Hint⁷ \triangleleft

Exercise 3.11. Buckling of a beam

Consider a horizontal beam located between 0 and 1 on the x -axis of the plane. We assume that the beam is fixed at $x = 0$ and $x = 1$ and that a force F is applied at $(1, 0)$ in the direction towards the origin. This situation can be modeled by the two point boundary value problem

$$u''(t) = -Ku(t), \quad u(0) = u(1) = 0, \quad K := \frac{F}{R}. \quad (3.5)$$

where $u(t)$ is the vertical displacement of the beam at t , and R is a constant depending from the rigidity of the beam. Clearly $u = 0$ is a solution, but we can have nonzero solutions, known as **eigenfunctions** corresponding to certain values of K known as **eigenvalues**. If $F = 0$ then $u = 0$ is the only solution, but if the force is increased it will reach a critical value where the beam will buckle and maybe break. This critical value corresponds to the smallest eigenvalue. In this exercise we

⁵ Recall the trigonometric formulas $\sin 2A = 2 \sin A \cos A$ and $\sin(A + B) + \sin(A - B) = 2 \sin A \cos B$.

⁶ Sum the geometric series.

⁷ $\mathbf{s}_j^T \mathbf{s}_j = \sum_{k=1}^m \sin^2(kj\pi h) = \frac{1}{2} \sum_{k=0}^m (1 - \cos(2kj\pi h))$. Then use 6.

will approximate (3.5) by a matrix eigenvalue problem. Using the finite difference approximation (see Chap. 13 for details)

$$u''(x) \approx \frac{u(x+h) - 2u(x) + u(x-h)}{h^2}$$

we obtain

$$\frac{-v_{j-1} + 2v_j - v_{j+1}}{h^2} = Kv_j, \quad j = 1, \dots, m, \quad h = \frac{1}{m+1}, \quad v_0 = v_{m+1} = 0,$$

where $v_j \approx u(jh)$ for $j = 0, \dots, m+1$. If we define $\lambda := h^2 K$ then we obtain the matrix eigenvalue problem

$$\mathbf{T}_m \mathbf{v} = \lambda \mathbf{v}, \text{ with } \mathbf{v} = [v_1, \dots, v_m]^T. \quad (3.6)$$

where \mathbf{T}_m was defined in Exercise 3.10 with its eigenvalues.

1. Show that the value of \tilde{K} corresponding to the smallest eigenvalue of \mathbf{T}_m is given by

$$\tilde{K} = \frac{2}{h^2} (1 - \cos(\pi h)).$$

2. It can be shown that the smallest eigenvalue of (3.5) is $K = \pi^2$. Show that $0 < K - \tilde{K} < \frac{\pi^4}{12} h^2$. \triangleright Math Hint⁸
3. To check the upper bound write a MATLAB program that for $h = 2^{-k}/10$, $k = 1 : 5$ computes $\pi^2 - \tilde{K}$ numerically using the MATLAB function `eig`. How does the error constant compare to the upper bound $\pi^4/12$?

3.5 Solutions

3.5.1 Elementary Computations

Exercise 3.1

- $\det(\mathbf{A} - t\mathbf{I}) = \begin{vmatrix} 2-t & 1 \\ 1 & 2-t \end{vmatrix} = t^2 - 4t + 3 = (1-t)(3-t)$; the eigenvalues are $\lambda_1 = 1$ with corresponding eigenspace $E_1 = \{[x, y]^T : x + y = 0\}$ with a direction vector $\mathbf{u}_1 = [1, -1]^T$ and $\lambda_2 = 3$ with $E_2 = \{x - y = 0\}$ and $\mathbf{u}_2 = [1, 1]^T$. Let $\mathbf{P} := [\mathbf{u}_1 \mathbf{u}_2] = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix}$, then $\mathbf{D} := \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix} = \mathbf{P}^{-1} \mathbf{A} \mathbf{P}$.
- \mathbf{I} is a diagonal matrix and the eigenvalues are the elements on the diagonal. There is only one eigenvalue, $\lambda = 1$ and its multiplicity is 2. The eigenspace is \mathbb{R}^2 .

⁸ $1 - x^2/2 < \cos x < 1 - x^2/2 + x^4/24$ for $x \in (0, \pi)$.

$$\text{c. } \det(\mathbf{B} - t\mathbf{I}) = \begin{vmatrix} 5-t & 3 & -3 \\ 1 & 3-t & 1 \\ 0 & 0 & 2-t \end{vmatrix} = (2-t)(t^2 - 8t + 12) = -(t-2)^2(t-6)$$

There are two eigenvalues $\lambda_1 = 2$ and $\lambda_2 = 6$.

i. $\lambda_1 = 2$: its multiplicity is 2. If $\mathbf{u} = [x, y, z]^T$ is an eigenvector then

$$\left\{ \begin{array}{lcl} 5x + 3y - 3z & = & 2x \\ x + 3y + z & = & 2y \\ 2z & = & 2z \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} 3x + 3y - 3z & = & 0 \\ x + y + z & = & 0 \\ z & = & 0 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} x + y & = & 0 \\ z & = & 0 \end{array} \right.$$

which gives an eigenspace of dimension 1 with direction vector for example, $\mathbf{u} = [1, -1, 0]^T$.

ii. $\lambda_2 = 6$: its multiplicity is 1, hence the dimension of the eigenspace is 1. If $\mathbf{v} = [x, y, z]^T$ is an eigenvector then

$$\left\{ \begin{array}{lcl} 5x + 3y - 3z & = & 6x \\ x + 3y + z & = & 6y \\ 2z & = & 6z \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} x - 3y & = & 0 \\ z & = & 0 \end{array} \right.$$

An eigenvector is $\mathbf{v} = [3, 1, 0]^T$.

$$\text{d. } \det(\mathbf{C} - t\mathbf{I}) = \begin{vmatrix} 1-t & 0 & 0 \\ 1 & 3-t & -1 \\ 1 & 2 & -t \end{vmatrix} = (1-t)(t^2 - 3t + 2) = -(t-1)^2(t-2)$$

There are two eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 2$.

i. $\lambda_1 = 1$: its multiplicity is 2. If $\mathbf{u} = [x, y, z]^T$ is an eigenvector then

$$\left\{ \begin{array}{lcl} x & = & x \\ x + 3y - z & = & y \\ x + 2y & = & z \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} x + 2y & = & z \end{array} \right.$$

which gives an eigenspace of dimension 2 with basis $\{\mathbf{u}_1, \mathbf{u}_2\}$ where, for example, $\mathbf{u}_1 = [1, 0, 1]^T$ and $\mathbf{u}_2 = [2, -1, 0]^T$.

ii. $\lambda_2 = 2$: its multiplicity is 1, hence the dimension of the eigenspace is 1. If $\mathbf{v} = [x, y, z]^T$ is an eigenvector then

$$\left\{ \begin{array}{lcl} x & = & 2x \\ x + 3y - z & = & 2y \\ x + 2y & = & 2z \end{array} \right. \Leftrightarrow \left\{ \begin{array}{lcl} x & = & 0 \\ y & = & z \end{array} \right.$$

An eigenvector is $\mathbf{v} = [0, 1, 1]^T$.

If $\mathbf{P} = \begin{bmatrix} 1 & 2 & 0 \\ 0 & -1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$, then $\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{bmatrix} = \mathbf{P}^{-1} \mathbf{A} \mathbf{P}$.

2. If $m - t = 1 \Leftrightarrow t = m - 1$, then $\mathbf{M} - t\mathbf{I} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ whose rank is 1. Thus $\ker(\mathbf{M} - t\mathbf{I})$ has dimension: $3 - \text{rank}(\mathbf{M}) = 2$. We deduce that $\lambda_1 = m - 1$ is an eigenvalue of \mathbf{M} with eigenspace of dimension 2 and multiplicity at least 2. The other eigenvalue satisfies $2\lambda_1 + \lambda_2 = \text{trace}(\mathbf{M}) = 3m$ which gives $\lambda_2 = m + 2$.
3. Computing $\det(\mathbf{J} - t\mathbf{I})$, we deduce that the only eigenvalue is $\lambda = 1$ with multiplicity 2 and the eigenspace is of dimension 1. If \mathbf{J} is similar to a diagonal matrix, then it is similar to \mathbf{I} , the identity matrix and the eigenspace for the eigenvalue 1 is \mathbb{R}^2 whose dimension is 2. This gives a contradiction.
4. $\det(\mathbf{J}_n - t\mathbf{I}) = (\lambda - t)^n$ so that the only eigenvalue of \mathbf{J}_n is λ with multiplicity n . If $\mathbf{u} = [x_1, \dots, x_n]^T$ is an eigenvector, then for $i = 1, \dots, n - 1$, at row i of $\mathbf{J}\mathbf{u} = \lambda\mathbf{u}$, we obtain $\lambda x_i + x_{i+1} = \lambda x_i$, thus $x_{i+1} = 0$. All components of \mathbf{u} are 0 except the first one since $\mathbf{u} \neq \mathbf{0}$. Thus $\mathbf{u} = x_1 \mathbf{e}_1$.

Exercise 3.2

$$\begin{aligned} \det(\mathbf{M} - t\mathbf{I}) &= \begin{vmatrix} 2-t & 1 & 3 & 4 & 5 \\ -1 & 2-t & 1 & 2 & 3 \\ 0 & 0 & 1-t & 2 & 3 \\ 0 & 0 & 0 & 3-t & 2 \\ 0 & 0 & 0 & -1 & 1-t \end{vmatrix} \\ &= \begin{vmatrix} 2-t & 1 \\ -1 & 2-t \end{vmatrix} \times (1-t) \times \begin{vmatrix} 3-t & 2 \\ -1 & 1-t \end{vmatrix} \\ &= (2-t+i)(2-t-i)(1-t)(t^2 - 4t + 5) \\ &= -(t-2-i)^2(t-2+i)^2(t-1) \end{aligned}$$

The eigenvalues are $\lambda_1 = 2 + i$, $\lambda_2 = \bar{\lambda}_1 = 2 - i$ both with multiplicity 2 and $\lambda_3 = 1$ with multiplicity 1.

Exercise 3.3

If λ is an eigenvalue of \mathbf{A} with eigenvector \mathbf{u} , then $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$. We multiply by \mathbf{A}^{-1} so that $\mathbf{u} = \lambda\mathbf{A}^{-1}\mathbf{u}$. With this equation, since $\mathbf{u} \neq \mathbf{0}$, we obtain that $\lambda \neq 0$. When dividing by λ , we deduce that $\frac{1}{\lambda}\mathbf{u} = \mathbf{A}^{-1}\mathbf{u}$ which proves that $1/\lambda$ is an eigenvalue of \mathbf{A}^{-1} with the eigenvector \mathbf{u} .

Exercise 3.4

Since \mathbf{A} is symmetric, its eigenvalues are real. Let λ be one of them with the eigenvector \mathbf{u} . We multiply $\mathbf{A}\mathbf{u} = \lambda\mathbf{u}$ by \mathbf{u}^T on the left and obtain $\mathbf{u}^T \mathbf{A} \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u}$. Now since $\mathbf{u} \neq \mathbf{0}$, we know that $\mathbf{u}^T \mathbf{u} > 0$ and $\mathbf{u}^T \mathbf{A} \mathbf{u} > 0$. Thus $\lambda > 0$.

Exercise 3.5

```
>> A=[0 ,1/4 ,3/4;3/4 ,0 ,1/4;1/4 ,3/4 ,0]
[P,D]=eig(A)
A =
      0     0.2500    0.7500
    0.7500        0    0.2500
    0.2500    0.7500        0
P =
    0.5774        -0.2887 - 0.5000 i   -0.2887 + 0.5000 i
    0.5774        -0.2887 + 0.5000 i   -0.2887 - 0.5000 i
    0.5774         0.5774        0.5774
D =
    1.0000        0        0
      0    -0.5000 + 0.4330 i        0
      0        0    -0.5000 - 0.4330 i
>> abs(D(2,2))
ans =
    0.6614
```

$|d_{22}| = |d_{33}| \simeq 0.6614 < 1$ so that $\lim_{n \rightarrow +\infty} D^n = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$. Since $A = PDP^{-1}$, we deduce that $A^n = P D^n P^{-1}$ and $\lim_{n \rightarrow +\infty} A^n = P \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} P^{-1}$

```
>> P*[1 ,0 ,0;0 ,0 ,0;0 ,0 ,0]*P^(-1)
ans =
      0.3333    0.3333    0.3333
      0.3333    0.3333    0.3333
      0.3333    0.3333    0.3333
>> A^100
ans =
      0.3333    0.3333    0.3333
      0.3333    0.3333    0.3333
      0.3333    0.3333    0.3333
```

3.5.2 The Power and Inverse Power Methods**Exercise 3.6**

- power1.m

```
maxiter=50;
A=rand(10);
x=rand(10,1);
q=x/norm(x,2);
for i=1:maxiter
```

```

x=A*q;
q=x/norm(x,2);
mu=q'*A*q;
end
error=norm(A*q-mu*q,2)

```

2. power2.m

```

poweral;
d=eig(A);
[1,i]=max(abs(d));
lambda=d(i)
disp('mu-lambda = ')
muend=lambda

```

3. power3.m

```

maxiter=20;
A=rand(10);
x=rand(10,1);
q=x/norm(x,2);
for i=1:maxiter
    x=A*q;
    q=x/norm(x,2);
    mu(i)=q'*A*q;
end
muend=mu(maxiter)
error=norm(A*q-muend*q,2)
d=eig(A);
[1,i]=max(abs(d));
lambda=d(i)
disp('mu-lambda = ')
muend-lambda
plot(1:maxiter, mu, '--')
xlabel('iteration')
ylabel('mu')

```

4. power4.m

```

maxiter=100;
A=[1,1,-2; 0,1,-1; 0,0 -0.99];
%A=[1,1,-2; 0,-1,-1; 0,0 0.1];
x=rand(3,1);
q=x/norm(x,2);
for i=1:maxiter
    x=A*q;
    q=x/norm(x,2);
    mu=q'*A*q;
end
error=norm(A*q-mu*q,2)

```

Exercise 3.7

```

function [s1,q1]=invpower(A,s,q)
x=(A-s*eye(length(A)))\q;

```

```
| q1=x/norm(x,2);
s1=q1'*A*q1;
```

3.5.3 The QR Method

Exercise 3.8

1. Since $A^T A = R^T R$, we obtain

$$Q^T Q = (R^{T^{-1}} A^T) A R^T = R^{T^{-1}} (A^T A) R^T = R^{T^{-1}} (R^T R) R^T = I$$

thus Q is a unitary matrix.

2. Answer:

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \left(\frac{1}{\sqrt{5}} \begin{bmatrix} 2 & -1 \\ 1 & 2 \end{bmatrix} \right) \times \left(\frac{1}{\sqrt{5}} \begin{bmatrix} 5 & 4 \\ 0 & 3 \end{bmatrix} \right) = QR.$$

3.

```
>> [Q,R]=qr([1,0,1;-2,-1,0;2,2,1])
Q =
    -0.3333    0.6667   -0.6667
    0.6667   -0.3333   -0.6667
   -0.6667   -0.6667   -0.3333
R =
    -3      -2      -1
     0      -1      0
     0       0      -1
```

Exercise 3.9

1.

```
>> A=[2,1;1,2]
A =
    2      1
    1      2
>> [Q,R]=qr(A); A=R*Q
A =
    2.8000   -0.6000
   -0.6000    1.2000
>> [Q,R]=qr(A); A=R*Q
A =
    2.9756    0.2195
    0.2195    1.0244
>> [Q,R]=qr(A); A=R*Q
A =
    2.9973   -0.0740
   -0.0740    1.0027
...
```

```
>> B=[5,3,-3;1,3,1;0,0,2];
>> [Q,R]=qr(B); B=R*Q
```

```
B =
5.6923 -2.4615 2.7456
-0.4615 2.3077 1.5689
0 0 2.0000
.....
>> [Q,R]=qr(B); B=R*Q

B =
5.9999 2.0002 -2.5299
0.0002 2.0001 1.8972
0 0 2.0000
```

Comment: We find the eigenvalues on the diagonal. ☺

```
>> [Q,R]=qr(C); C=R*Q

C =
2.3333 0.3563 2.4689
0.8909 1.2381 1.6496
-0.3086 -0.0825 0.4286
...
>> [Q,R]=qr(C); C=R*Q

C =
2.0000 -0.3333 3.2998
0.0000 1.0000 0.0000
-0.0000 0.0000 1.0000
```

2.

```
>> D=[0,-1,0; 1,0,0; 0,0,1];
>> [Q,R]=qr(D); D=R*Q
```

```
D =
0 1 0
-1 0 0
0 0 1
```

```
>> [Q,R]=qr(D); D=R*Q
```

```
D =
0 -1 0
1 0 0
0 0 1
```

Comment: The basic QR method does not converge. ☹

3.5.4 Application to the Buckling of a Beam

Exercise 3.10

1. If $m = 2$, then $h = 1/3$ and $\cos(1\pi h) = 1/2 = -\cos(2\pi h)$.
Since $\mathbf{T}_2 = \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$, we deduce $\det(\mathbf{T}_2 - t\mathbf{I}) = (2-t)^2 - 1 = (1-t)(3-t)$.
The eigenvalues are $\lambda_1 = 1 = 2(1 - \cos(1\pi h))$ and $\lambda_2 = 3 = 2(1 - \cos(2\pi h))$.
2. The formulas are a direct consequence of the trigonometric formulas $\sin 2A = 2 \sin A \cos A$ and $\sin(A+B) + \sin(A-B) = 2 \sin A \cos B$.
3. For $j = 1, 2$, let $\mathbf{s}_j := [\sin(j\pi/3), \sin(2j\pi/3)]^T \neq 0$. Then

$$\mathbf{T}_2 \mathbf{s}_j = \begin{bmatrix} 2 \sin(j\pi/3) - \sin(2j\pi/3) \\ -\sin(j\pi/3) + 2 \sin(2j\pi/3) \end{bmatrix} = \begin{bmatrix} 2(1 - \cos(j\pi/3)) \sin(j\pi/3) \\ 2(1 - \cos(j\pi/3)) \sin(2j\pi/3) \end{bmatrix}$$

where for the first row, $A = j\pi/3$ and $2 \sin A - \sin 2A = 2(1 - \cos(A)) \sin A$ and for the second one, $A = 2j\pi/3$, $B = j\pi/3$ thus $\sin(A+B) = 0$ then $-\sin(A-B) + 2 \sin A - \sin(A+B) = 2(1 - \cos(B)) \sin A$. This proves that \mathbf{s}_j is an eigenvector of \mathbf{T}_2 for the eigenvalue $2(1 - \cos(j\pi/3))$.

Moreover $\mathbf{s}_1^T \mathbf{s}_2 = [\sqrt{3}, \sqrt{3}] \times \begin{bmatrix} \sqrt{3} \\ -\sqrt{3} \end{bmatrix} / 4 = 0$ while $\mathbf{s}_1^T \mathbf{s}_1 = [\sqrt{3}, \sqrt{3}] \times \begin{bmatrix} \sqrt{3} \\ \sqrt{3} \end{bmatrix} / 4 = 3/2 = 1/(2h)$ and also $\mathbf{s}_2^T \mathbf{s}_2 = 1/(2h)$.

4. For $m \geq 2$ and $j = 1, \dots, m$, let $\mathbf{s}_j = [\sin(kj\pi h)]_{k=1,\dots,m} \in \mathbb{R}^m$. Then $\mathbf{s}_j \neq 0$ and the k -th row of $\mathbf{T}_m \mathbf{s}_j$ is

$$-\sin((k-1)j\pi h) + 2 \sin(kj\pi h) - \sin((k+1)j\pi h) = 2(1 - \cos(j\pi h)) \sin(kj\pi h)$$

using again $-\sin(A-B) + 2 \sin A - \sin(A+B) = 2(1 - \cos(B)) \sin A$ with $A = kj\pi h$ and $B = j\pi h$. This also includes the first and last row since in these cases $\sin((k-1)j\pi h) = 0$ for $k = 1$ and $\sin((k+1)j\pi h) = 0$ for $k = m$.

5. We have proved that \mathbf{s}_j is an eigenvector of \mathbf{T}_m for the eigenvalue $2(1 - \cos(j\pi h))$. Now, \mathbf{T}_m is a symmetric matrix thus has an orthogonal basis of m eigenvectors. We have obtained m different eigenvalues and the corresponding eigenspaces are of dimension 1. Hence the eigenvectors \mathbf{s}_j are orthogonal, that is $\mathbf{s}_j^T \mathbf{s}_\ell = 0$ for $j \neq \ell$.
6. For $j = 1, \dots, m$, $e^{2ij\pi h} \neq 1$ since $0 < 2j\pi/(m+1) < 2\pi$. We deduce that for the geometric sequence $(e^{2ij\pi h})^k$,

$$\sum_{k=0}^m e^{2ikj\pi h} = \frac{1 - (e^{2ij\pi h})^{m+1}}{1 - e^{2ij\pi h}} = 0 \text{ since } (e^{2ij\pi h})^{m+1} = e^{i2j\pi} = 1.$$

Then we also have $\sum_{k=0}^m \cos 2ikj\pi h = \sum_{k=0}^m \sin 2ikj\pi h = 0$.

7. We compute

$$\begin{aligned}\mathbf{s}_j^T \mathbf{s}_j &= \sum_{k=1}^m \sin^2(kj\pi h) = \sum_{k=0}^m \sin^2(kj\pi h) = \frac{1}{2} \sum_{k=0}^m (1 - \cos(2kj\pi h)) \\ &= \frac{m+1}{2} - \frac{1}{2} \sum_{k=0}^m \cos(2kj\pi h) = \frac{m+1}{2} = 1/(2h).\end{aligned}$$

Exercise 3.11

1. In the previous exercise, we have seen that the eigenvalues of T_m are $\lambda_j = 2(1 - \cos(j\pi h))$ for $j = 1, \dots, m$. The smallest one is obtained for the maximal value of $\cos j\pi h$ which is obtained for $j = 1$. Thus, the corresponding $\tilde{K} = \lambda_1/h^2 = \frac{2}{h^2}(1 - \cos(\pi h))$.
2. By Taylor expansions around 0, for any $x \in (0, \pi/2)$, there exist $\theta_1, \theta_2 \in (0, 1)$ such that $\cos x = 1 - x^2/2 \cos(\theta_1 x) > 1 - x^2/2$ and $\cos x = 1 - x^2/2 + x^4/24 \cos(\theta_2 x) < 1 - x^2/2 + x^4/24$. If $K = \pi^2$, then $K - \tilde{K} = \pi^2 - \frac{2}{h^2}(1 - \cos(\pi h))$. With the previous inequalities, we obtain

$$\begin{aligned}\pi^2 - \frac{2}{h^2}((\pi h)^2/2) &< K - \tilde{K} < \pi^2 - \frac{2}{h^2}((\pi h)^2/2 - (\pi h)^4/24) \\ \Leftrightarrow 0 &< K - \tilde{K} < \pi^4 h^2/12.\end{aligned}$$

3. beameuigenv.m

```
arrk = 1:7;
for k=1:length(arrk)
    h=2^(-k)/10;
    tK=2/h^2*(1-cos(pi*h));
    arrerr(k)=abs(tK-pi^2)/h^2;
end
plot(arrk , arrerr , 'x--' , arrk , pi^4/12*ones(length(arrk),1))
xlabel('k')
```

Also see Fig. 3.2.

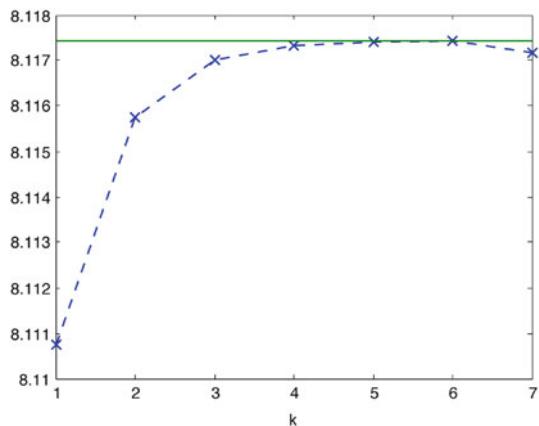


Fig. 3.2 $(K - \tilde{K})/h^2$ compared to $\pi^4/12$

Chapter 4

Matrices, Norms and Conditioning

To measure the size of vector and matrices we use norms. In this chapter, we introduce and study three vector norms and define and use the corresponding matrix norms. One of the goals is to estimate a bound on the error in the solution \mathbf{x} of a linear system $A\mathbf{x} = \mathbf{b}$ when there exists an uncertainty in the exact values of the data A , and \mathbf{b} .

The first section is devoted to direct computations of norms, with examples in Exercises 4.1 and 4.2 and with special cases of the spectral norm in Exercise 4.3 and of a Hermitian (symmetric) matrix in Exercise 4.4.

Two important inequalities involve condition numbers of linear systems. In Exercises 4.5 and 4.6, we recall the proof of these and show with an example that the inequality can become equality. In Exercise 4.7, we study the condition number of a Vandermonde matrix as a function of the dimension.

The next two exercises, Exercises 4.8 and 4.9 show that the condition number is a notion somewhat different from eigenvalues or determinants.

Exercise 4.10 study the conditioning of an important matrix that we have already found in Exercise 3.10 and that is also used in Chap. 13. With Exercise 4.11, we prove that for any nonsingular matrix and the spectral norm, it is always possible to transform the inequality obtained in Exercise 4.5 into an equality.

Finally, in Exercise 4.12, we study a least-squares problem with a special matrix. More least-squares problems are studied in Chap. 10.

Review: To a vector norm $\|\cdot\|$ on \mathbb{R}^n or \mathbb{C}^n we associate the corresponding **matrix norm** on $\mathbb{R}^{m \times n}$ or $\mathbb{C}^{m \times n}$ by

$$\|A\| := \max_{\mathbf{x} \neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|} = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\|.$$

This **operator norm** satisfies the **consistency condition**, $\|AB\| \leq \|A\| \|B\|$ for any rectangular matrices such that the product AB is defined.

We will use the following vector norms and corresponding matrix norms¹:

1. $\|\mathbf{x}\|_1 := \sum_{j=1}^n |x_j|$, $\|\mathbf{A}\|_1 = \max_{1 \leq j \leq n} \sum_{k=1}^m |a_{k,j}|$, **(one-norm)**,
2. $\|\mathbf{x}\|_2 := \sqrt{\sum_{j=1}^n |x_j|^2}$, $\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A}^* \mathbf{A})}$, **(two-norm, spectral norm)**, where $\rho(\mathbf{A}^* \mathbf{A})$ is the largest eigenvalue of $\mathbf{A}^* \mathbf{A}$. If \mathbf{A} is Hermitian then $\|\mathbf{A}\|_2 = \rho(\mathbf{A})$, the **spectral radius** of \mathbf{A} , i.e., the largest eigenvalue of \mathbf{A} in absolute value.
3. $\|\mathbf{x}\|_\infty := \max_{1 \leq j \leq n} |x_j|$, $\|\mathbf{A}\|_\infty = \max_{1 \leq k \leq m} \sum_{j=1}^n |a_{k,j}|$, **(infinity-norm)**,

In this chapter, unless otherwise specified, $\|\mathbf{A}\|$ will denote one of $\|\mathbf{A}\|_p$ for $p = 1, 2, \infty$. For a nonsingular $\mathbf{A} \in \mathbb{C}^{n \times n}$ the product

$$\text{cond}(\mathbf{A}) := \|\mathbf{A}\| \|\mathbf{A}^{-1}\|,$$

is called a **condition number** (with respect to inversion). We note that

1. For the spectral norm $\|\mathbf{U} \mathbf{A} \mathbf{V}\|_2 = \|\mathbf{A}\|_2$ for any square \mathbf{A} and square unitary matrices \mathbf{U}, \mathbf{V} .
2. If \mathbf{A} is nonsingular then $\|\mathbf{A}^{-1}\|_2 = 1/\sigma$, where σ is the nonnegative square root of the smallest eigenvalue of $\mathbf{A}^* \mathbf{A}$.

¶

4.1 Elementary Examples

Exercise 4.1. First examples

1. Find the $1, 2, \infty$ norms of the matrices

$$\mathbf{T} := \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}, \quad \mathbf{I} := \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{J} := \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Then find their p condition numbers $\text{cond}_p(\mathbf{T})$ for $p = 1, 2, \infty$.

2. In Exercise 2.4 we have shown that $\mathbf{AB} = \mathbf{I}$ where

¹ More generally we have the p vector norms given for $p \geq 1$ by $\|\mathbf{x}\|_p := \left(\sum_{j=1}^n |x_j|^p \right)^{1/p}$. We have $\lim_{p \rightarrow \infty} \|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty$.

$$\mathbf{A} := \begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix}, \quad \mathbf{B} := \begin{bmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-\ell} & \dots & (-2)^{n-\ell} \\ \vdots & & & \ddots & \ddots & \ddots & \vdots \\ 0 & & & & 0 & 1 & -2 \\ \dots & & & & \dots & 0 & 1 \end{bmatrix}$$

Determine $\|\mathbf{A}\|_1$, $\|\mathbf{A}^{-1}\|_1$ and $\text{cond}_1(\mathbf{A})$.

Exercise 4.2. Bounding the spectral norm

If $\mathbf{A} \in \mathbb{C}^{m \times n}$, we recall that $\|\mathbf{A}\|_2^2 = \rho(\mathbf{A}^* \mathbf{A})$. We want to show the inequality

$$\|\mathbf{A}\|_2^2 \leq \|\mathbf{A}\|_1 \|\mathbf{A}\|_\infty. \quad (4.1)$$

1. Verify the inequality for the matrices in Exercise 4.1, Question 1.
2. Let $\lambda := \rho(\mathbf{A}^* \mathbf{A})$ and \mathbf{v} a corresponding eigenvector. Show that $\|\mathbf{A}\|_2^2 \|\mathbf{v}\|_1 = \|\mathbf{A}^* \mathbf{A} \mathbf{v}\|_1$.
3. Deduce the inequality (4.1).

Exercise 4.3. Spectral radius

Show that for any vector norm and corresponding matrix norm, $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$ for any $\mathbf{A} \in \mathbb{R}^{n \times n}$ where $\rho(\mathbf{A}) = \max\{|\lambda| : \lambda \in \sigma(\mathbf{A})\}$ is the spectral radius of \mathbf{A} .
 ▷ Math Hint² ◷

Exercise 4.4. Spectral norm of an Hermitian matrix

Let \mathbf{A} be Hermitian (i.e. $\mathbf{A}^* = \mathbf{A}$, see Chap. 3) with largest (resp. smallest) eigenvalue λ_1 (resp. λ_n) in absolute value. Show that $\|\mathbf{A}\|_2 = |\lambda_1|$. If in addition \mathbf{A} is nonsingular then $\|\mathbf{A}^{-1}\|_2 = 1/|\lambda_n|$.

4.2 Conditioning and Error

Exercise 4.5. Perturbation in right-hand-side

Let \mathbf{b} and $\delta\mathbf{b}$ be two vectors in \mathbb{R}^n (or \mathbb{C}^n) with $\mathbf{b} \neq \mathbf{0}$, and $\mathbf{A} \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$) a nonsingular matrix. Let \mathbf{x} and $\mathbf{x} + \delta\mathbf{x}$ be the solutions of $\mathbf{Ax} = \mathbf{b}$ and $\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$.

1. Show that $\|\mathbf{b}\| \leq \|\mathbf{A}\| \|\mathbf{x}\|$ and $\|\delta\mathbf{x}\| \leq \|\mathbf{A}^{-1}\| \|\delta\mathbf{b}\|$.
2. Conclude that

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}. \quad (4.2)$$

² Let $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, where $\mathbf{v} \neq \mathbf{0}$. Take norms and use the consistency condition.

Comment: This shows that the relative error in the solution is bounded by the condition number times the relative error in the right-hand-side. ☺

3. We can have equality in (4.2). Consider the following example suggested by R.S. Wilson and considered in [18]

$$A = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}, \quad b = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix}, \quad \delta b = \begin{bmatrix} 0.01 \\ -0.01 \\ 0.01 \\ -0.01 \end{bmatrix}.$$

- a. Solve the systems $Ax = b$ and $A\delta x = \delta b$ using MATLAB.
- b. Compute numerically $\|\delta x\|_\infty / \|x\|_\infty$, $\text{cond}_\infty(A)$ and $\text{cond}_\infty(A)\|\delta b\|_\infty / \|b\|_\infty$.
- c. Same questions with $\|\cdot\|_2$.

Exercise 4.6. Perturbation of matrix elements

Suppose $A, A + \delta A \in \mathbb{R}^{n \times n}$ (or $\mathbb{C}^{n \times n}$) are nonsingular matrices and $b \in \mathbb{R}^n$ (or \mathbb{C}^n) with $b \neq 0$. If $Ax = b = (A + \delta A)(x + \delta x)$ then prove that

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \text{cond}(A) \frac{\|\delta A\|}{\|A\|}. \quad (4.3)$$

Exercise 4.7. Condition number of a Vandermonde matrix

For $x_n = [x_1, \dots, x_n]^T$ the corresponding Vandermonde matrix is defined by $V_n = V(x_1, \dots, x_n) := [x_i^{j-1}]_{i,j=1,\dots,n}$.

Alexandre-Théophile Vandermonde (1735–1796) was a French musician (violinist), mathematician and chemist. His name is associated with determinant theory in mathematics. He became engaged in mathematics only around 1770 and was elected to the French Academy of Sciences. In his mathematical work, the Vandermonde determinant does not appear explicitly.



1. Prove that $\det V_n = \prod_{1 \leq i < j \leq n} (x_j - x_i)$ and show that if the x_i are all distinct then V is nonsingular. ▷ Math Hint³ ◁
2. For a given positive integer n , we define $x_n = [1/n, 2/n, \dots, 1]^T \in \mathbb{R}^n$. Write a MATLAB function `condvdm.m` with input n and output the condition number of Vandermonde matrix V_n corresponding to x_n using $\|\cdot\|_1$.

```
>> c=condvdm(3)
c =
    72.0000
```

3. For different values of n (for example n from 5 to 15) write a MATLAB program `plotcondvdm.m` that plots $\log(\text{cond}(V_n))$ as a function of n and finds an approximation of $\text{cond}(V_n)$ as a function of n . See Fig. 4.1.

³ Use induction on $n = k$. To show the result for $n = k + 1$ define the polynomial π_k by $\pi_k(x) = \det V(x, x_2, \dots, x_{k+1})$. Show that $\pi_k(x) = c_k \prod_{j=2}^{k+1} (x_j - x)$, where $c_k = \det V(x_2, \dots, x_{k+1})$.

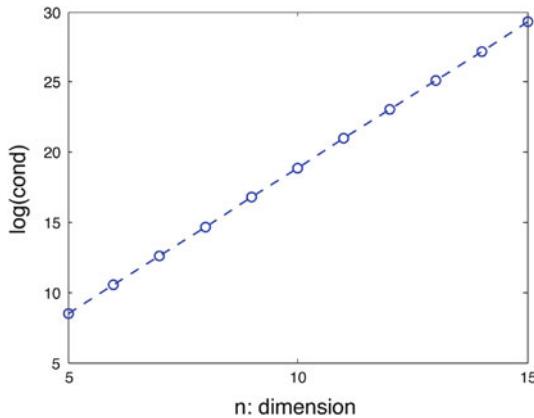


Fig. 4.1 Conditioning number of a Vandermonde matrix

4.3 Conditioning, Eigenvalues and Determinant

Exercise 4.8. A poorly conditioned system

We define for $n \geq 0$ a function $f : \mathbb{R}^{n+1} \rightarrow \mathbb{R}$ by

$$f(\mathbf{x}) = f(x_0, \dots, x_n) = \int_0^1 (x_0 + x_1 t + \dots + x_n t^n)^2 dt,$$

1. Show that $f(\mathbf{x}) = \mathbf{x}^T \mathbf{H} \mathbf{x}$ where $\mathbf{H} = [\frac{1}{1+i+j}]_{i,j=0}^n \in \mathbb{R}^{(n+1) \times (n+1)}$.
2. Show that \mathbf{H} is symmetric positive definite. \triangleright Math Hint⁴ \triangleleft
3. Show that if λ is an eigenvalue of \mathbf{H} then λ is real and

$$\frac{1}{\|\mathbf{H}^{-1}\|} \leq \lambda \leq \|\mathbf{H}\|. \quad (4.4)$$

4. In this part we use MATLAB.

- a. For a fixed n construct \mathbf{H} (without using loops \triangleleft MATLAB Hint⁵ \triangleright) and compute \mathbf{H}^{-1} . Save as `hilbmatrix1.m`. Test with $n = 3$.

```
>> hilbmatrix
H =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
```

⁴ For positive definite, use $f(\mathbf{x}) = \mathbf{x}^T \mathbf{H} \mathbf{x}$.

⁵ For a vector \mathbf{x} , use `x*ones(n+1, 1)`.

```
invH =
1.0e+03 *
0.0160   -0.1200    0.2400   -0.1400
-0.1200    1.2000   -2.7000    1.6800
 0.2400   -2.7000    6.4800   -4.2000
-0.1400    1.6800   -4.2000    2.8000
```

- b. Compute numerically $\|\mathbf{H}\|_\infty$ and $\|\mathbf{H}^{-1}\|_\infty$. Save as hilbmatrix2.m. Test with $n = 5$.

```
>> hilbmatrix2
n =
5
norm of H:
nH =
2.4500
norm of inv(H):
ninvH =
1.1865e+07
```

- c. Give upper and lower bounds for the eigenvalues of \mathbf{H} using (4.4) and compute the ∞ condition number of \mathbf{H} . Save as hilbmatrix3.m. Test with $n = 5$.

```
>> hilbmatrix3
n =
5
Conditionning number of H
ans =
2.9070e+07
```

- d. To solve $\mathbf{H}\mathbf{x} = \mathbf{b}$ for $n = 5$, suppose we only know \mathbf{b} with approximately 5 significant digits, i.e., $\frac{\|\delta\mathbf{b}\|_\infty}{\|\mathbf{b}\|_\infty} \leq 10^{-5}$. With how many significant digits are we guaranteed to know the exact solution \mathbf{x} ? \triangleright Math Hint⁶ \triangleleft

Comment: The matrix \mathbf{H} is known as the **Hilbert matrix** and is an example of a poorly conditioned matrix. ☺

David Hilbert (1862–1943) was a German mathematician known as one of the founders of proof theory and mathematical logic. He is one of the most influential and universal mathematicians of the 19th and early 20th centuries. He worked on invariant theory, on the axiomatization of geometry and on functional analysis. In 1900, he presented various problems that set the course for much of the mathematical research of the 20th century.



⁶ Use (4.2).

Exercise 4.9. Conditioning and determinant

Let \mathbf{E} et \mathbf{U} be square matrices of order n and $\mathbf{f} \in \mathbb{R}^n$ be defined by

$$\mathbf{E} = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & & 0 \\ 0 & 1 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{U} = \begin{bmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ 0 & 1 & \cdots & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix},$$

and let

$$\mathbf{A} = \begin{bmatrix} 2 & 2 & \cdots & \cdots & 2 \\ -1 & 1 & 1 & \cdots & 1 \\ 0 & -1 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

1. We have $\mathbf{A} = \mathbf{U} - \mathbf{E} + e_1 \mathbf{f}^T$, where $e_1 = [1, 0, \dots, 0]^T \in \mathbb{R}^n$. Verify this for $n = 4$.
2. We also have $\mathbf{A} = (2\mathbf{I} - \mathbf{E})\mathbf{U}$. Show this for $n = 4$. Then find $\det(\mathbf{A})$ for general n .
3. Find $\mathbf{E}^2, \mathbf{E}^3, \dots, \mathbf{E}^n$.
4. Show that $\mathbf{U}^{-1} = \mathbf{I} - \mathbf{E}^T$ and that $\left(\mathbf{I} - \frac{\mathbf{E}}{2}\right)^{-1} = \mathbf{I} + \frac{\mathbf{E}}{2} + \frac{\mathbf{E}^2}{4} + \dots + \frac{\mathbf{E}^{n-1}}{2^{n-1}}$.
5. Find \mathbf{A}^{-1} and show that $\|\mathbf{A}^{-1}\|_\infty = 1 - \frac{1}{2^n}$.
6. Conclude that $\text{cond}_\infty(\mathbf{A}) \sim 2n$.

Comment: The value of the determinant is 2^n and we see that the determinant cannot in general be used to estimate the condition number of a matrix.

**Exercise 4.10. Condition number of 2. derivative matrix**

In this exercise, for $m \geq 2$, we determine the spectral condition number $\text{cond}_2(\mathbf{T}_m) = \|\mathbf{T}_m\|_2 \|\mathbf{T}_m^{-1}\|_2$ of the tridiagonal matrix

$$\mathbf{T}_m := \begin{bmatrix} 2 & -1 & 0 & & & \\ -1 & 2 & -1 & & & \\ 0 & \ddots & \ddots & \ddots & & 0 \\ & & & & -1 & 2 & -1 \\ & & & & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{m \times m} \quad (4.5)$$

and give upper and lower bounds showing how it grows with m . Since \mathbf{T}_m is symmetric it follows from Exercise 4.4 that $\text{cond}_2(\mathbf{T}_m) = |\lambda_1|/|\lambda_m|$, where λ_1 (resp. λ_m) is the largest (resp. smallest) eigenvalue in absolute value of \mathbf{T}_m . The

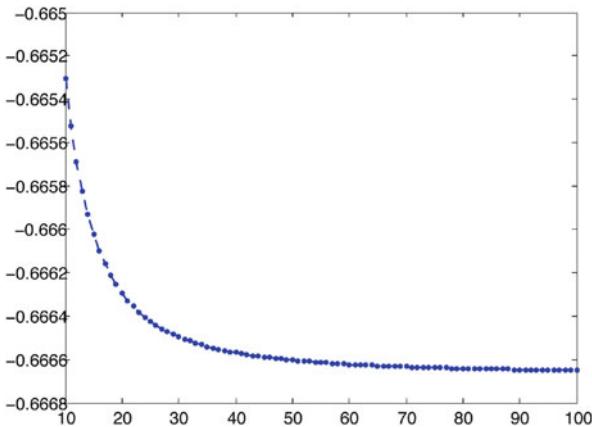


Fig. 4.2 Computation of $\text{cond}_2(\mathbf{T}_m) - 4(m+1)^2/\pi^2$

eigenvalues of \mathbf{T}_m have been computed in Exercise 3.10 and we recall that they are given by $\lambda_j = 2(1 - \cos(j\pi h))$, for $j = 1, \dots, m$, where $h := 1/(m+1)$.

1. Show that

$$\text{cond}_2(\mathbf{T}_m) = \cot^2\left(\frac{\pi h}{2}\right) = 1/\tan^2\left(\frac{\pi h}{2}\right).$$

▷ *Math Hint*⁷ ◁

2. Write a MATLAB program that for a given value of m computes the corresponding matrix \mathbf{T}_m without using loops and then compute its 2-norm condition number. Save as `condTm.m`. Test with $m = 10$.

```
>> condTm
m =
    10
cm =
   48.3742
```

3. Write a second MATLAB program that for an array $arrm$ containing given values of m compute the corresponding matrices \mathbf{T}_m and condition numbers $c_m = \text{cond}_2(\mathbf{T}_m)$ and then plots the graph of $c_m - (m+1)^2 * 4/\pi^2$. Save as `limcondTm.m`. Test with $arrm = 10 : 100$.

© **Comment:** It can be shown that the following bounds hold

$$\frac{4}{\pi^2}(m+1)^2 - \frac{2}{3} < \text{cond}_2(\mathbf{T}_m) < \frac{4}{\pi^2}(m+1)^2, \quad m \geq 2. \quad (4.6)$$

In Fig. 4.2 the limit value seems to be $-2/3$ in agreement with the lower bound in (4.6). ☺

⁷ If $t = \tan(\theta/2)$ then $\cos \theta = \frac{1-t^2}{1+t^2}$.

Exercise 4.11. Equality in (4.2) with the spectral norm

1. Using MATLAB construct a random matrix $A \in \mathbb{R}^{10 \times 10}$ and compute the eigenvalues of $S := A^T A$. Show mathematically that the eigenvalues of S are nonnegative. Compute the largest eigenvalue λ and the smallest eigenvalue μ of S . Compare $\|A\|_2$ and $\|A^{-1}\|_2$ with $\sqrt{\lambda}$ and $1/\sqrt{\mu}$. Make several tests.

```
>> eigenvandspectraln

eigenvalues =
    0.0357      0.0656      0.2955      0.3187      0.5075
    1.0698      1.4391      1.4857      2.1923      23.8451

lmax =
    23.8451

lmin =
    0.0357

normA =
    4.8831

sqrtlambda =
    4.8831

normAm1 =
    5.2958

oneoversqrmu =
    5.2958
```

©Comment: This verifies numerically that $\|A\|_2 = \sqrt{\lambda}$ and $\|A^{-1}\|_2 = 1/\sqrt{\mu}$, cf. Exercise 4.4 ☺

2. With the help of the previous matrix A or a new random matrix

- Construct $b := Ax$, where $x := [1, \dots, 1]^T$.
- Generate a random vector δb and solve for δx in $A\delta x = \delta b$.
- Numerically, compare $\frac{\|\delta b\|_2}{\|b\|_2}$, $\frac{\|\delta x\|_2}{\|x\|_2}$ and $\|A\|_2 \|A^{-1}\|_2 \frac{\|\delta b\|_2}{\|b\|_2}$.

```
>> errorandcond

rb =
    0.1518

rx =
    1.5758

ans =
    7.1815
```

```

cond A norm(db)/norm(b) - norm(dx)/norm(x)

ans =
5.6058

```

⊕Comment: This verifies numerically the inequality (4.2). ⊕

3. The worst is coming . . . , the case of equality.

We continue with the matrix $S := \mathbf{A}^T \mathbf{A}$ with largest eigenvalue λ . Let \mathbf{u} be an eigenvector corresponding to λ and (μ, \mathbf{v}) an eigenpair corresponding to the smallest eigenvalue μ of S . We define $\mathbf{b} := \mathbf{A}\mathbf{u}$ and $\delta\mathbf{b} := \mathbf{A}\mathbf{v}$. Consider (4.2) with $\mathbf{x} = \mathbf{u}$ and $\delta\mathbf{x} = \mathbf{v}$.

Compare $\frac{\|\delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$ and $\|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2 \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}$ ◁ MATLAB Hint⁸ ▷ Test on several examples.

```

>> condequality

arrlambda =
    0.0202    0.0204    0.0701    0.1716    0.5859
    0.9408    1.0502    1.6755    2.3919    25.866

rb =
    0.0279

ru =
    1.0000

cond A norm(db)/norm(b) - norm(dx)/norm(x)

ans =
3.3307e-16

```

4. Prove that in this case we have the equality

$$\frac{\|\delta\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \text{cond}_2(\mathbf{A}) \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}.$$

Exercise 4.12. A weighted least squares problem We want to approximate a given function $f \in C[0, 1]$ by a sequence (p_n) of polynomials of degree n . For each n we minimize

$$E(p) = \int_0^1 (f(t) - p(t))^2 \sqrt{t} dt, \quad p \in \mathbb{P}_n.$$

If $p(x) := \sum_{i=0}^n a_i x^i$, the minimum $p = p_n$ is obtained when $\frac{\partial E}{\partial a_i} = 0$ for $i = 0, \dots, n$ giving the system $\mathbf{M}\mathbf{a} = \mathbf{b}$, where

⁸ One can use the instruction `[V, D] = eig(S)` and `[C, I] = max(...)` giving the position `I` of the maximum.

$$\mathbf{M} = [m_{ij}]_{i,j=0}^n, \text{ with } m_{ij} = \int_0^1 t^{i+j+1/2} dt = \frac{1}{i+j+3/2},$$

$$\mathbf{b} = (b_i)_{i=0}^n, \text{ with } b_i = \int_0^1 f(t)t^{i+1/2} dt,$$

$$\mathbf{a} = [a_0, a_1, \dots, a_n]^T, \text{ coefficients of } p_n.$$

1. Write a program to evaluate the function $f1(\mathbf{x}) = \sin \pi \mathbf{x}$ for a vector \mathbf{x} and save in `f1.m`. Compute `f1([1/4, -1])`.

```
>> f1 ([1/4 , -1])
ans =
    0.7071      -0.0000
```

2. Generate the matrix $\mathbf{M} \in \mathbb{R}^{(n+1) \times (n+1)}$ without using loops \triangleleft MATLAB Hint⁹ \triangleright and save in `approx1.m`. Test with $n = 4$.

```
>> approx1
M =
    0.6667    0.4000    0.2857    0.2222    0.1818
    0.4000    0.2857    0.2222    0.1818    0.1538
    0.2857    0.2222    0.1818    0.1538    0.1333
    0.2222    0.1818    0.1538    0.1333    0.1176
    0.1818    0.1538    0.1333    0.1176    0.1053
```

3. Construct \mathbf{b} by using the files `sndpart.m` and `functj.m` below; The call is `b=sndpart('f', n)` where `f` is the name of the file containing the function. The files allows one to calculate an approximation to the integral with the help of the function `quadl.m`. Test with `f=f1` and $n = 4$.

```
function y=functj(x,namefunc,j);
y=feval(namefunc,x).*x.^^(j+1/2);
```

```
function b=sndpart(namefunc,n);
for i=1:n+1
    b(i)=quadl('functj',0,1,[ ],[ ],namefunc,i-1);
end;
```

```
>> approx2
b =
    0.4374      0.2416      0.1521      0.1041      0.0755
```

4. Once the solution \mathbf{a} is computed, we can compute the polynomial with the help of `polyval` (pay attention to the ordering of the coefficients). Calculate \mathbf{a} and plot the graph of f and the polynomial p_n . Save in `approx3.m`. Test with $n = 4$ and `f1`.

⁹ See Exercise 4.8.

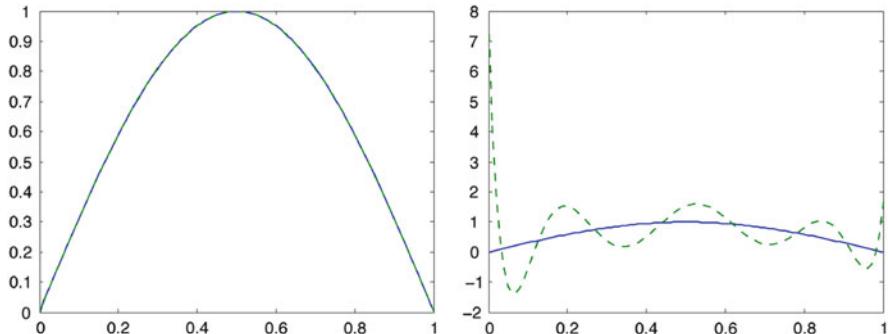


Fig. 4.3 Function and polynomial approximation for $n = 4$ on the *left* and $n = 8$ on the *right*

```
>> approx3
Coefficients of the polynomial
a =
    0.0022
    3.0765
    0.5720
   -7.2921
    3.6428
```

5. Try different values of n , say $n = 2, 3, 5, 10, 15, 20, \dots$ and the functions f_1, f_2, f_3 with $f_1(x) = \sin \pi x$, $f_2(x) = \sin 4\pi x$, $f_3(x) = |x - 1/2|$. See Fig. 4.3. What happens when $n \geq 20$?

4.4 Solutions

4.4.1 Elementary Examples

Exercise 4.1

- $\|\mathbf{T}\|_1 = \|\mathbf{T}\|_\infty = 3$. The eigenvalues of \mathbf{T} are 1 and 3. Since \mathbf{T} is symmetric we find $\|\mathbf{T}\|_2 = \rho(\mathbf{T}) = 3$. Now $\mathbf{T}^{-1} = \frac{1}{3} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$ and $\|\mathbf{T}^{-1}\|_1 = \|\mathbf{T}^{-1}\|_\infty = 1$. The eigenvalues of \mathbf{T}^{-1} are 1 and $1/3$ so that $\|\mathbf{T}^{-1}\|_2 = 1$. We deduce that $\text{cond}_1(\mathbf{T}) = \text{cond}_2(\mathbf{T}) = \text{cond}_\infty(\mathbf{T}) = 3$.
- $\|\mathbf{I}\|_1 = \|\mathbf{I}\|_\infty = \|\mathbf{I}\|_2 = 1$. Since $\mathbf{I}^{-1} = \mathbf{I}$, we conclude that $\text{cond}_1(\mathbf{I}) = \text{cond}_2(\mathbf{I}) = \text{cond}_\infty(\mathbf{I}) = 1$.
- $\|\mathbf{J}\|_1 = \|\mathbf{J}\|_\infty = 2$. The eigenvalues of $\mathbf{J}^T \mathbf{J} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$ are $\frac{3 \pm \sqrt{5}}{2}$ so that $\|\mathbf{J}\|_2 = \sqrt{\frac{3 + \sqrt{5}}{2}}$. $\mathbf{J}^{-1} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$ and $\|\mathbf{J}^{-1}\|_1 = \|\mathbf{J}^{-1}\|_\infty = 2$. The

eigenvalues of $\mathbf{J}^{-1T}\mathbf{J}^{-1} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$ are $\frac{3 \pm \sqrt{5}}{2}$ therefore $\|\mathbf{J}^{-1}\|_2 = \sqrt{\frac{3 + \sqrt{5}}{2}} = \|\mathbf{J}\|_2$. Let us notice that $\frac{3 + \sqrt{5}}{2} = \frac{2}{3 - \sqrt{5}}$ so that the eigenvalues are inverses of each other. To conclude, $\text{cond}_1(\mathbf{J}) = \text{cond}_\infty(\mathbf{J}) = 4$ while $\text{cond}_2(\mathbf{J}) = \frac{3 + \sqrt{5}}{2}$.

2. $\|A\|_1 = 3$, $\|A^{-1}\|_1 = \sum_{k=0}^{n-1} 2^k = 2^n - 1$ and $\text{cond}_1(A) = 3(2^n - 1)$.

Exercise 4.2

1. $\|\mathbf{T}\|_1 = \|\mathbf{T}\|_\infty = \|\mathbf{T}\|_2 = 3$, thus $\|\mathbf{T}\|_2^2 = \|\mathbf{T}\|_1 \|\mathbf{T}\|_\infty$ and we have a similar equality for \mathbf{I} .

$\|\mathbf{J}\|_1 = \|\mathbf{J}\|_\infty = 2$ while $\|\mathbf{J}\|_2^2 = \frac{3 + \sqrt{5}}{2} < 3 < \|\mathbf{J}\|_1 \|\mathbf{J}\|_\infty$.

2. $\lambda := \rho(\mathbf{A}^* \mathbf{A}) = \|\mathbf{A}\|_2^2$ and $\mathbf{A}^* \mathbf{A} \mathbf{v} = \lambda \mathbf{v}$. Thus

$$\|\mathbf{A}\|_2^2 \|\mathbf{v}\|_1 = \lambda \|\mathbf{v}\|_1 = \|\lambda \mathbf{v}\|_1 = \|\mathbf{A}^* \mathbf{A} \mathbf{v}\|_1 \leq \|\mathbf{A}^*\|_1 \|\mathbf{A}\|_1 \|\mathbf{v}\|_1.$$

3. Observing that $\|\mathbf{A}^*\|_1 = \|\mathbf{A}\|_\infty$ and canceling $\|\mathbf{v}\|_1$ since \mathbf{v} is a nonzero vector, proves the result.

Exercise 4.3

Let λ be an eigenvalue of \mathbf{A} and let \mathbf{v} be a corresponding eigenvector. Then $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$, so that $|\lambda|\|\mathbf{v}\| = \|\lambda\mathbf{v}\| = \|\mathbf{A}\mathbf{v}\| \leq \|\mathbf{A}\|\|\mathbf{v}\|$. Since $\mathbf{v} \neq \mathbf{0}$, we deduce that $|\lambda| \leq \|\mathbf{A}\|$ and conclude that $\rho(\mathbf{A}) \leq \|\mathbf{A}\|$

Exercise 4.4

Let \mathbf{A} be a Hermitian matrix. Then the eigenvalues are real. Moreover, $\mathbf{A}^* \mathbf{A} = \mathbf{A}^2$ and the eigenvalues of \mathbf{A}^2 are the squares of the eigenvalues of \mathbf{A} . It follows that $\|\mathbf{A}\|_2 = \sqrt{\rho(\mathbf{A})^2} = \rho(\mathbf{A})$. Since the eigenvalues of \mathbf{A}^{-1} are the inverses of the eigenvalues of \mathbf{A} it follows that $\|\mathbf{A}^{-1}\|_2 = 1/|\lambda_n|$, where λ_n is the smallest eigenvalue of \mathbf{A} in absolute value.

4.4.2 Conditioning and Error

Exercise 4.5

1. We first note that $\|\mathbf{b}\| = \|\mathbf{Ax}\| \leq \|\mathbf{A}\|\|\mathbf{x}\|$. By subtracting $\mathbf{Ax} = \mathbf{b}$ from $\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) = \mathbf{b} + \delta\mathbf{b}$ we deduce that $\mathbf{A}\delta\mathbf{x} = \delta\mathbf{b}$ or equivalently $\delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{b}$. But then $\|\delta\mathbf{x}\| = \|\mathbf{A}^{-1}\delta\mathbf{b}\| \leq \|\mathbf{A}^{-1}\| \|\delta\mathbf{b}\|$.
2. By multiplication of the two previous inequalities, we obtain

$$\|\mathbf{b}\| \|\delta\mathbf{x}\| \leq \|\mathbf{A}\| \|\mathbf{x}\| \|\mathbf{A}^{-1}\| \|\delta\mathbf{b}\|$$

from which we deduce (4.2).

3. a. With MATLAB

```
>> A=[10 7 8 7;7 5 6 5;8 6 10 9;7 5 9 10];
>> b=[32 23 33 31]'; db=[.01 -.01 .01 -.01]';
>> x=A\b; dx=A\db;
```

we obtain $\mathbf{x} = [1 \ 1 \ 1]^T$ and $\delta\mathbf{x} = [0.82 \ -1.36 \ 0.35 \ 0.21]^T$

b. We find $\|\mathbf{b}\|_\infty = 33$, $\|\delta\mathbf{b}\|_\infty = 0.01$, $\|\mathbf{x}\|_\infty = 1$, $\|\delta\mathbf{x}\|_\infty = 1.36$. Moreover,

$\det(\mathbf{A}) = 1$ and $\mathbf{A}^{-1} = \begin{bmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{bmatrix}$. Thus $\|\delta\mathbf{x}\|_\infty/\|\mathbf{x}\|_\infty =$

1.36, $\text{cond}_\infty(\mathbf{A}) = 33 \times 136 = 4488$ and

$$\|\delta\mathbf{x}\|_\infty/\|\mathbf{x}\|_\infty = \text{cond}_\infty(\mathbf{A})\|\delta\mathbf{b}\|_\infty/\|\mathbf{b}\|_\infty = 1.36.$$

c.

```
>> norm(dx, 2)/norm(x, 2)
ans =
0.8198
>> cond(A, 2)
ans =
2.9841e+03
>> cond(A, 2)* norm(db, 2)/norm(b, 2)
ans =
0.9943
```

Exercise 4.6

Since $\mathbf{b} = \mathbf{A}\mathbf{x} = (\mathbf{A} + \delta\mathbf{A})(\mathbf{x} + \delta\mathbf{x})$, we obtain

$$\mathbf{b} = \mathbf{b} + \mathbf{A}\delta\mathbf{x} + \delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}) \Rightarrow \delta\mathbf{x} = \mathbf{A}^{-1}\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x}).$$

We deduce that

$$\|\delta\mathbf{x}\| = \|\mathbf{A}^{-1}\delta\mathbf{A}(\mathbf{x} + \delta\mathbf{x})\| \leq \|\mathbf{A}^{-1}\| \|\delta\mathbf{A}\| \|\mathbf{x} + \delta\mathbf{x}\| = \text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} \|\mathbf{x} + \delta\mathbf{x}\|,$$

and the result follows.

Exercise 4.7

1. We will prove by induction the property for $k \geq 2$

$$P_k : \forall \mathbf{x} \in \mathbb{R}^k, d_k := \det(\mathbf{V}_k) = \prod_{1 \leq i < j \leq k} (x_j - x_i).$$

For $n = 2$, $\mathbf{V}_2 = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix}$ and $d_2 = (x_2 - x_1)$ so that (P_2) is proved.

Define the polynomial π_k by $\pi_k(x) = \det V(x, x_2, \dots, x_{k+1})$. When expanding the determinant in terms of the first row, we see that π_k is a polynomial of degree not greater than k . Moreover $\pi_k(x_j) = 0$ for $j = 2, \dots, k+1$ since in case $x = x_j$, two rows of the matrix are duplicated. Thus $\pi_k(x) = c_k \prod_{j=2}^{k+1} (x_j - x)$ for some constant c_k . To find c_k we compute the x^k coefficient of $\pi_k(x)$ in two ways.

On one hand, with the expansion of $\pi_k(x)$, this coefficient is $c_k(-1)^k$, and on the other hand, when expanding the determinant of $V(x, x_2, \dots, x_{k+1})$ in terms of the first row, we obtain the x^k coefficient $(-1)^k \det V(x_2, \dots, x_{k+1})$. By the induction hypothesis we have $c_k = \det V(x_2, \dots, x_{k+1}) = \prod_{2 \leq i < j \leq k+1} (x_j - x_i)$. Setting $x = x_1$, we obtain P_{k+1} .

2. condvdm.m

```
function c=condvdm(n)
x=(1:n)'/n;
for j=0:n-1
    V(:,j+1)=x.^j;
end
c=cond(V,1);
```

3. plotcondvdm.m

```
arrn=5:10;
for k=1:length(arrn)
    arrc(k)=condvdm(arrn(k));
end
plot(arrn,log(arrc),'o—')
xlabel('n: dimension')
ylabel('log(cond)')
a=polyfit(arrn,log(arrc),1)
```

```
>> plotcondvdm
a =
    2.0826    -1.9596
```

So that the conditioning $c_n \approx e^{a(1)n+a(2)} \approx 0.1409e^{2.0826n}$.

4.4.3 Conditioning, Eigenvalues and Determinant

Exercise 4.8

- If $H = [h_{ij}]_{i,j=0}^n$, for a given $\mathbf{x} \in \mathbb{R}^n$,

$$\mathbf{x}^T H \mathbf{x} = \sum_{i=0}^n \sum_{j=0}^n h_{ij} x_i x_j$$

and

$$\begin{aligned} f(\mathbf{x}) &= \int_0^1 \left(\sum_{i=0}^n x_i t^i \right) \left(\sum_{j=0}^n x_j t^j \right) dt = \sum_{i=0}^n \sum_{j=0}^n x_i x_j \int_0^1 t^{i+j} dt \\ &= \sum_{i=0}^n \sum_{j=0}^n x_i x_j \frac{1}{i+j+1}. \end{aligned}$$

So that $f(\mathbf{x}) = \mathbf{x}^T H \mathbf{x}$ for any \mathbf{x} as soon as $h_{ij} = \frac{1}{i+j+1}$.

2. Clearly $h_{ij} = \frac{1}{i+j+1} = h_{ji}$ and $\mathbf{x}^T H \mathbf{x} = f(\mathbf{x}) = \int_0^1 p_n(t)^2 dt \geq 0$, where $p_n(t) = x_0 + x_1 t + \dots + x_n t^n$. If $\mathbf{x}^T H \mathbf{x} = 0$ then $p_n^2(t) = 0$ for all $t \in [0, 1]$ since p_n^2 is continuous, nonnegative and satisfies $\int_0^1 p_n^2(t) dt = 0$. Hence $p_n = 0$ on $[0, 1]$ so that every x_i is 0 or equivalently $\mathbf{x} = 0$.
3. The matrix \mathbf{H} is symmetric so that its eigenvalues are real. If λ is an eigenvalue of \mathbf{H} with corresponding eigenvector $\mathbf{u} \in \mathbb{R}^{n+1}$ then $\mathbf{H}\mathbf{u} = \lambda\mathbf{u}$, from which we obtain $\mathbf{u}^T \mathbf{H} \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u}$. Now $\mathbf{u}^T \mathbf{H} \mathbf{u} > 0$ since \mathbf{H} is positive definite and $\mathbf{u}^T \mathbf{u} = \|\mathbf{u}\|_2^2 > 0$, hence $\lambda > 0$. Now

$$\mathbf{H}\mathbf{u} = \lambda\mathbf{u} \Rightarrow \|\lambda\mathbf{u}\| = \lambda\|\mathbf{u}\| = \|\mathbf{H}\mathbf{u}\| \leq \|\mathbf{H}\|\|\mathbf{u}\| \Rightarrow \lambda \leq \|\mathbf{H}\|.$$

and similarly

$$\mathbf{H}\mathbf{u} = \lambda\mathbf{u} \Rightarrow \mathbf{H}^{-1}\mathbf{u} = \frac{1}{\lambda}\mathbf{u} \Rightarrow \frac{1}{\lambda} \leq \|\mathbf{H}^{-1}\|$$

and (4.4) follows.

4. a. hilbmatrix1.m

```
n=3;
x=0:n;
Hr=x'*ones(1,n+1);
H=1./(Hr+Hr'+1)
invH=inv(H)
```

- b. hilbmatrix2.m

```
n=5
x=0:n;
Hr=x'*ones(1,n+1);
H=1./(Hr+Hr'+1);
invH=inv(H);
disp('norm of H: ')
nH=norm(H, inf)
disp('norm of inv(H): ')
ninvH=norm(invH, inf)
```

- c. hilbmatrix3.m

```
n=5
x=0:n;
Hr=x'*ones(1,n+1);
H=1./(Hr+Hr'+1);
invH=inv(H);
disp('norm of H: ')
nH=norm(H, inf)
disp('norm of inv(H): ')
ninvH=norm(invH, inf)
invninvH=1/ninvH
disp('Conditionning number of H')
cond(H, inf)
```

From (4.4), we obtain that the eigenvalues satisfy $1/1.1865 \times 10^7 = 8.4279 \times 10^{-8} \leq \lambda \leq 2.4500$.

- d. We suppose that $\frac{\|\delta b\|_\infty}{\|b\|_\infty} \leq 10^{-5}$, then $\frac{\|\delta x\|_\infty}{\|x\|_\infty} \leq \text{cond}(\mathbf{H}) \times 10^{-5} \simeq 3 \times 10^3$ which gives no real garancy on the relevance of the solution x of the system $Hx = b$...

Exercise 4.9

1. For $n = 4$

$$\begin{aligned} \mathbf{U} - \mathbf{E} + \mathbf{e}_1 \mathbf{f}^T &= \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 2 & 2 & 2 & 2 \\ -1 & 1 & 1 & 1 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \end{aligned}$$

2.

$$(2\mathbf{I} - \mathbf{E})\mathbf{U} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{A}.$$

From $\mathbf{A} = (2\mathbf{I} - \mathbf{E})\mathbf{U}$, we find $\det(\mathbf{A}) = \det(2\mathbf{I} - \mathbf{E}) \det \mathbf{U} = 2^n$ since we have a product of triangular matrices, and the determinant of a triangular matrix is the product of the diagonal elements.

3.

$$\begin{aligned} \mathbf{E} &= \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & & 0 \\ 0 & 1 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}, \quad \mathbf{E}^2 = \begin{bmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & & 0 \\ 1 & 0 & 0 & & \vdots \\ 0 & \ddots & \ddots & \ddots & \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \dots, \\ \mathbf{E}^{n-1} &= \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & & & \vdots \\ 0 & 0 & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix} \end{aligned}$$

and finally \mathbf{E}^n is the null matrix. In the computation of the successive powers, the nonzero sub-diagonal goes down one level at each new power.

4. For $n = 4$

$$(\mathbf{I} - \mathbf{E}^T)\mathbf{U} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \mathbf{I}$$

so that $\mathbf{U}^{-1} = \mathbf{I} - \mathbf{E}^T$.

Then $(\mathbf{I} - \frac{\mathbf{E}}{2}) \times \left(\mathbf{I} + \frac{\mathbf{E}}{2} + \frac{\mathbf{E}^2}{4} + \dots + \frac{\mathbf{E}^{n-1}}{2^{n-1}} \right) = \mathbf{I} - \frac{\mathbf{E}^n}{2^n} = \mathbf{I}$ and we obtain
 $\left(\mathbf{I} - \frac{\mathbf{E}}{2} \right)^{-1}$.

5. Since $\mathbf{A} = 2(\mathbf{I} - \mathbf{E}/2) \times \mathbf{U}$, we deduce that

$$\mathbf{A}^{-1} = \frac{1}{2} \mathbf{U}^{-1} \times (\mathbf{I} - \mathbf{E}/2)^{-1} = \frac{1}{2} (\mathbf{I} - \mathbf{E}^T) \times \left(\mathbf{I} + \frac{\mathbf{E}}{2} + \frac{\mathbf{E}^2}{4} + \dots + \frac{\mathbf{E}^{n-1}}{2^{n-1}} \right).$$

So that

$$\mathbf{A}^{-1} = \frac{1}{2} \begin{bmatrix} 1/2 & -1 & 0 & 0 & \dots & 0 \\ 1/4 & 1/2 & -1 & 0 & \dots & 0 \\ 1/8 & 1/4 & 1/2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 1/2^{n-1} & 1/2^{n-2} & \dots & \dots & 1/2 & -1 \\ 1/2^{n-1} & 1/2^{n-2} & \dots & \dots & 1/2 & 1 \end{bmatrix}$$

We compute $\|\mathbf{A}^{-1}\|_\infty = \frac{1}{2} \sum_{k=0}^{n-1} 2^{-k} = 1 - 1/2^n$ and then $\text{cond}_\infty(\mathbf{A}) = 2n.(1 - 2^{-n}) \approx 2n$.

Exercise 4.10

1. With the known eigenvalues, the smallest one is $\lambda_1 = 2(1 - \cos(\pi h))$ and the largest one is $\lambda_m = 2(1 - \cos(\pi mh)) = 2(1 - \cos(\pi(m+1)h - \pi h)) = 2(1 + \cos(\pi h))$. We recall that if $t = \tan(\theta/2)$ then $\cos \theta = \frac{1-t^2}{1+t^2}$. With $\theta = \pi h$, we obtain that

$$\text{cond}_2(\mathbf{T}_m) = \lambda_m / \lambda_1 = \frac{1 + \cos(\pi h)}{1 + \cos(\pi h)} = \frac{1 + \frac{1-t^2}{1+t^2}}{1 - \frac{1-t^2}{1+t^2}} = 1/t^2 = \cot^2\left(\frac{\pi h}{2}\right).$$

2. condTm.m

```
m=10
Mm=eye(m)-diag(ones(m-1,1),1);
Tm=Mm*Mm';
cm=cond(Tm,2)
```

3. limcondTm.m

```

arrm = 10:100;
for i=1:length(arrm)
    m=arrm(i);
    Mm=eye(m)-diag(ones(m-1,1),1);
    Tm=Mm*Mm';
    cm(i)=cond(Tm,2);
end
sqrmp1=(arrm+1).^2;
plot(arrm,cm-sqrmp1.*4/pi.^2,'.--')

```

Exercise 4.11

1. If $S = A^T A$ then $S^T = A^T (A^T)^T = A^T A = S$ therefore S is a symmetric matrix. Thus its eigenvalues are real. Moreover, if λ is one of these eigenvalues with associate eigenvector u , then $Su = \lambda u \Rightarrow u^T Su = \lambda u^T u$. Since $u \neq 0$, $u^T u = \|u\|_2^2 > 0$ and $u^T Su = (Au)^T (Au) = \|Au\|_2^2 > 0$. We deduce that $\lambda > 0$.

With the following program, we have a numerical check of the equality $\|A\|_2 = \sqrt{\lambda}$ where λ is the largest eigenvalue of $S = A^T A$ and a similar result for $\|A^{-1}\|_2 = 1/\mu$ where μ is the smallest eigenvalue of S .

eigenvandspectraln.m

```

A=rand(10);
S=A'*A;
Lambda=eig(S)
lmax=max(Lambda)
lmin=min(Lambda)
normA=norm(A,2)
sqrtlambda=sqrt(lmax)
normAml=norm(A^(-1),2)
oneoversqrtmu=1/sqrt(lmin)

```

2. With the following program, we check (numerically) the inequality (4.2).
errorandcond.m

```

A=rand(10);
x=ones(10,1);
b=A*x;
deltab=rand(10,1);
deltax=A\deltab;
nb=norm(b,2);
ndeltab=norm(deltab,2);
nx=norm(x,2);
ndeltax=norm(deltax,2);
rb=nodeltab/nb;
rx=nodeltax/nx;
cond(A,2)*rb;
disp('cond(A)*norm(db)/norm(b) - norm(dx)/norm(x)')
cond(A,2)*rb-rx

```

©Comment: In the examples, we also see that the relative error $\|\delta\mathbf{x}\|_2/\|\mathbf{x}\|_2$ is much greater than $\|\delta\mathbf{b}\|_2/\|\mathbf{b}\|_2$. ☺

3. Now we show numerically how (4.2) becomes an equality with an appropriate choice of \mathbf{b} and $\delta\mathbf{b}$ or equivalently \mathbf{x} and $\delta\mathbf{x}$.
condequality.m

```
A=rand(10);
S=A'*A;
[P,D]=eig(S);
arrlambda=max(D) % Array of the eigenvalues
[lambdamax,imax]=max(arrlambda); % max eigenvalue and position
[lambdamin,imin]=min(arrlambda); % min eigenvalue and position
u=P(:,imax); % vector x
v=P(:,imin); %vector delta x
b=A*u;
deltab=A*v;
nb=norm(b,2);
ndeltab=norm(deltab,2);
nu=norm(u,2);
ndeltau=norm(v,2);
rb=ndeltab/nb;
ru=ndeltau/nu
disp('cond A norm(db)/norm(b) - norm(dx)/norm(x)')
cond(A,2)*rb-ru
```

©Comment: We see that $\text{cond}(\mathbf{A}) \|\delta\mathbf{b}\|_2/\|\mathbf{b}\|_2 - \|\delta\mathbf{x}\|_2/\|\mathbf{x}\|_2 \approx 0$. ☺

4. Let us prove that the equality is always possible. We recall that $\sqrt{\lambda} = \rho(\mathbf{S})^{1/2} = \|\mathbf{A}\|_2$ and, similarly, $1/\sqrt{\mu} = \|\mathbf{A}^{-1}\|_2$. We have defined $\mathbf{x} = \mathbf{u}$ to be an eigenvector corresponding to λ and $\delta\mathbf{x} = \mathbf{v}$ an eigenvector corresponding to μ .

Now $\mathbf{A}^T \mathbf{A} \mathbf{u} = \lambda \mathbf{u}$ so that

$$\mathbf{u}^T \mathbf{A}^T \mathbf{A} \mathbf{u} = (\mathbf{A} \mathbf{u})^T (\mathbf{A} \mathbf{u}) = \|\mathbf{A} \mathbf{u}\|_2^2 = \lambda \mathbf{u}^T \mathbf{u} = \lambda \|\mathbf{u}\|_2^2.$$

Since $\mathbf{A} \mathbf{u} = \mathbf{b}$, we deduce that

$$\sqrt{\lambda} \|\mathbf{u}\|_2 = \|\mathbf{b}\|_2. \quad (4.7)$$

Similarly since $\mathbf{A}^T \mathbf{A} \mathbf{v} = \mu \mathbf{v}$ and $\mathbf{A} \mathbf{v} = \delta\mathbf{b}$, we obtain

$$\sqrt{\mu} \|\mathbf{v}\|_2 = \|\delta\mathbf{b}\|_2. \quad (4.8)$$

If we divide (4.8) by (4.7), we have

$$\frac{\sqrt{\mu} \|\mathbf{v}\|_2}{\sqrt{\lambda} \|\mathbf{u}\|_2} = \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2} \Leftrightarrow \frac{\|\mathbf{v}\|_2}{\|\mathbf{u}\|_2} = \frac{\sqrt{\lambda}}{\sqrt{\mu}} \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2} = \text{cond}(\mathbf{A}) \frac{\|\delta\mathbf{b}\|_2}{\|\mathbf{b}\|_2}.$$

Exercise 4.121. `f1.m`

```
function y=f1(x)
y=sin(pi*x);
```

2. `approx1.m`

```
n=5;
M=1./((0:n)'*ones(1,n+1)+ones(n+1,1)*(0:n)+3/2)
```

3. `approx2.m`

```
n=4;
namefunc='f1';
M=1./((0:n)'*ones(1,n+1)+ones(n+1,1)*(0:n)+3/2);
b=sndpart(namefunc,n)
```

4. `approx3.m`

```
n=4;
namefunc='f1';
M=1./((0:n)'*ones(1,n+1)+ones(n+1,1)*(0:n)+3/2);
b=sndpart(namefunc,n);
a=M\b';
disp('Coefficients of the polynomial')
a
x=[0:0.01:1];
y=feval(namefunc,x);
appr=polyval(a(length(a):-1:1),x);
plot(x,y,x,appr,'—');
```

② **Comment:** When n grows, the approximation is getting bad, which was impossible in theory since $\mathbb{P}_n \subset \mathbb{P}_{n+1}$. But, in fact, when n gets large, the matrix M is badly conditioned and MATLAB reports the fact. Moreover, the second member of the equation was approximated. These two reasons explain the large errors on the coefficients a and catastrophic polynomial approximations. ☺

Chapter 5

Iterative Methods

Iterative methods are used to approximate the solution of different kinds of linear or non-linear problems and also to construct curves (and surfaces).

In the first section, we deal with the solution of $f(x) = x$ using the fixed-point method in Exercises 5.1 and 5.2 where for the second one the rate of convergence to a solution is studied. In Exercises 5.3 and 5.4 we see that even for very simple functions f , there are many different ways to define the iterations.

Iterative methods can also be used to approximate the solution of a linear system. In Exercises 5.5–5.7 we study the Jacobi and Gauss-Seidel methods while in Exercises 5.9 and 5.10, the gradient method is described with an estimation of the rate of convergence.

In Exercise 5.11, we use a subdivision method to construct a C^1 interpolating function in \mathbb{R}^2 or \mathbb{R}^3 . This method was first published in 1986–1987. Subdivision is an iterative method.

Finally we consider an iterative method for a non-linear problem in Exercise 5.12.

Iterative methods are also used for many numerical problems and in other chapters of this book, for example

- To compute the eigenvalues of a matrix, (Exercises 3.5–3.7).
- To construct a Bézier curve with de Casteljau Algorithm 7.1.
- To approximate the value of an integral with the Monte Carlo method (Exercises 9.9 and 9.10).
- To compute the best approximation of a continuous function by a polynomial of given degree with the Remez algorithm (Exercise 11.6).
- To approximate the solution of differential equations in Chap. 12 in particular the shooting method (Exercise 12.12).
- ...

We also recover Gauss Seidel's method in Exercise 13.7.

5.1 Fixed-Point Methods

Review: Let I be a real interval. For a continuous function $\varphi : I \rightarrow I$, $x^* \in I$ is a **fixed-point** if $x^* = \varphi(x^*)$. We consider a sequence (x_k) given by

$$x_0 \in I, x_{k+1} = \varphi(x_k), k = 0, 1, 2, \dots \quad (5.1)$$

If the sequence x_n converges, then it converges to a fixed-point. If φ admits a derivative on I that satisfies $|\varphi'(x)| < 1$ for all $x \in I$, then there exists one and only one fixed-point x^* and the sequence x_n is well defined and converges to x^* for any starting value $x_0 \in I$. We define $e_n := x_n - x^*$ and say that the convergence is of order p if there exists a constant c such that $|e_{n+1}| \sim c|e_n|^p$ when n tends to $+\infty$.

Newton's method to find a root x^* of a differentiable function f is a fixed-point iteration with $\varphi(x) := x - \frac{f(x)}{f'(x)}$. If this method converges, the convergence is then of order $p = 2$. ♣

Exercise 5.1. A first example

$$\text{Let } \varphi_1(x) := \frac{1}{2} \left(x + \frac{2}{x} \right).$$

1. Show that $x^* = \sqrt{2}$ is a fixed-point for φ_1 . Are there other fixed-points?
2. Write a MATLAB function `e=squareroot2(x,n)` that computes the errors $e_k := x_k - \sqrt{2}$ for $k = 0, 1, 2, \dots, n$, starting with $x_0 = x$. Use `format short e`. What seems to be the order of the method?

```
>> e=squareroot2(2,5)
e =
  5.8579e-01
  8.5786e-02
  2.4531e-03
  2.1239e-06
  1.5947e-12
  -2.2204e-16
```

3. Show that the fixed-point method using φ_1 converges to $\sqrt{2}$ for any $x_0 \in I := (\sqrt{2}, \infty)$. ▷ *Math Hint*¹ ◁
4. Show that the method converges to $\sqrt{2}$ for any $x_0 > 0$. ▷ *Math Hint*² ◁
5. Show that if $x_k > 0$ then

$$x_{k+1} - \sqrt{2} = \frac{1}{2x_k} (x_k - \sqrt{2})^2, \quad k = 0, 1, 2, \dots \quad (5.2)$$

What is the order of convergence?

¹ Show that $\varphi_1 : I \rightarrow I$ and $0 < \varphi_1(x) < 1$ for all $x \in I$.

² Show that $x_1 > \sqrt{2}$ whenever $0 < x_0 < \sqrt{2}$.

6. Show that the fixed-point iteration with φ_1 is in fact Newton's method applied to $f(x) := x^2 - 2$, $x > 0$.

Comment: We have shown using fixed-point techniques that Newton's method to find a root of $x^2 - 2$ converges quadratically for any starting value $x_0 > 0$. ☺

Exercise 5.2. Numerical evaluation of the order

Let $\varphi_2(x) := \sin(x^a)$ where $a \geq 1$ is a real number.

- Find the fixed-points of φ_2 on $[0, 1]$ and show that $\varphi_2([0, 1]) \subset [0, 1]$.
- We start with $x_0 = 1$. Show that the corresponding sequence $x_{n+1} = \varphi_2(x_n)$ is decreasing and positive so that it converges.
- Programs: Write a MATLAB function file `phi2.m` that computes $\varphi_2(u, a)$ at a point u . Test:

```
>> phi2(1,3)
ans =
    0.8415
```

- Write a program that computes the iterates x_n . Stop the iteration when $n = maxiter$ or $|x_n - x_{n+1}| \leq precis$ where $maxiter$ and $precis$ are given. Display the last values $maxn := n$, the corresponding x_n (if the algorithm converges) and the $error := |x_n - x_{n-1}|$. ▷ MATLAB Hint³ ▷ Save as `fixpoint1.m`. Test with $a = 3$, $x_0 = 1$, $maxiter = 20$, $precis = 10^{-10}$.

```
>> fixpoint1
ans =
    7.0857e-62
number of iterations:
iter =
    8
```

- Add the graph of x_n as a function of n and save as `fixpoint2.m`. Same test.
- Evaluation of the order:** Modify the previous program to compute $err_n = |x_n - x_{nmax}|$ (we assume that x_{nmax} is a good approximation of the limit) then plot $\log(err_n)$ function of $\log(err_{n-1})$ for $n = 2, \dots, maxn - 1$. What seems to be the order? Save as `fixpoint3.m`. Test with $a = 1.3$, $a = 2.5$. See Fig. 5.1.
- Compute the exact order.

Exercise 5.3. Strange attractors

Let $\varphi_3(x) := \lambda x(1 - x)$ where λ is a real parameter.

- Show that for $\lambda \in [0, 4]$, we have $\varphi_3([0, 1]) \subset [0, 1]$.
- Find the fixed-points of φ_3 and those of $\varphi_3 \circ \varphi_3$
- Programs:** Write a MATLAB function file `phi3.m` that compute the values of the previous functions with inputs x and λ . Test:

```
>> phi3(1,0.5)
ans =
    0.8415
```

³ For the loop, use while (iter<maxiter) & (error>precis)

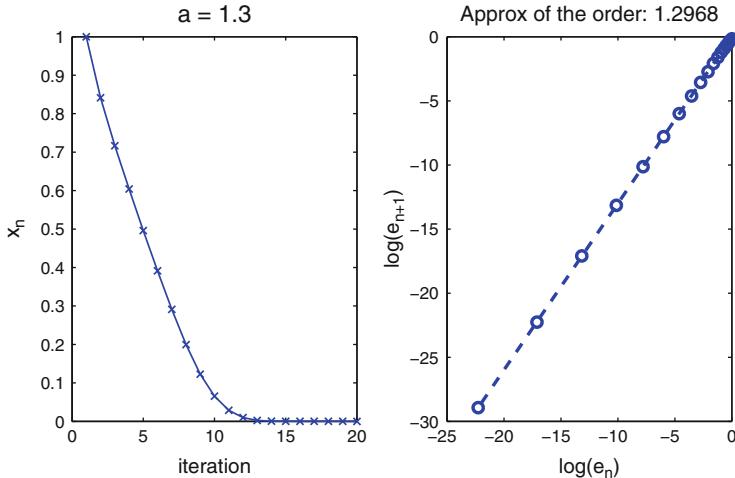


Fig. 5.1 Numerical evaluation of the order

4. Write a program that computes and plots the iterates x_n . Stop the iteration when $n = \text{maxiter}$ or $|x_n - x_{n+1}| \leq \text{precis}$ where maxiter and precis are given. Save as `newfixpoint.m`. Test $\text{maxiter} = 100$, $\text{precis} = 10^{-10}$.

- $\lambda = 0.5$, $x_0 = 0.1$,
- $\lambda = 1.5$, $x_0 = 0.1$,
- $\lambda = 3.1$, $x_0 = 0.1$.
- ...

Exercise 5.4. Fix point method in dimension 2, Henon's attractor

Michel Hénon (1931–2013) was a French mathematician and astronomer who worked at the Nice Observatory. He is known for his contributions to stellar dynamics and on the dynamical evolution of star clusters. In mathematics, he is well known for the Hénon map, a dynamical system that exhibits chaotic behavior.



Let $\varphi_4 \left(\begin{bmatrix} x \\ y \end{bmatrix} \right) = \begin{bmatrix} 1 - ax^2 + y \\ bx \end{bmatrix}$ where a, b are real parameters with $a \neq 0$.

1. Find the fixed-points of φ_4 .
 2. Write a program that computes and plots the iterates $\mathbf{u}_n = (x_n, y_n)$. Stop the iteration when $n = \text{maxiter}$ or $\|\mathbf{u}_n - \mathbf{u}_{n+1}\|_2 \leq \text{precis}$ where maxiter and precis are given. Save as `fixpointdim2.m`. Test with $\text{maxiter} = 100$ and $\text{precis} = 10^{-8}$ (Fig. 5.2).
- $a = 0.1, b = 0.3$. $(x_0, y_0) = (1, 1)$
 - $a = 0.5, b = 0.3$. periodic attraction
 - $a = 0.92, 1.03, 1.052$ etc....

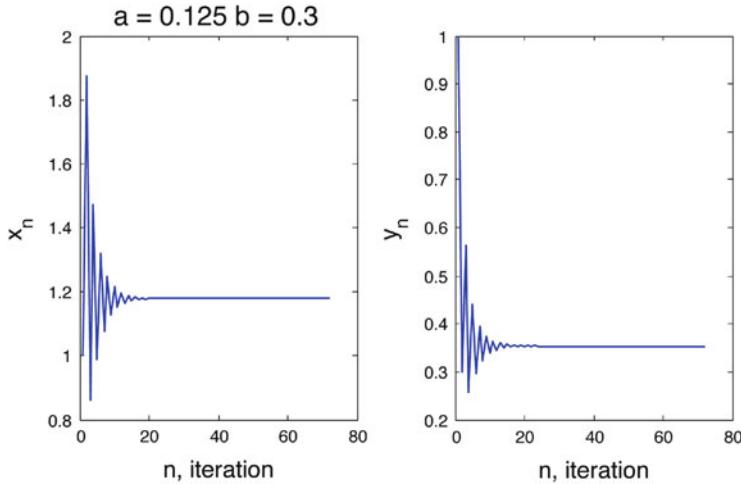


Fig. 5.2 The sequences x_n and y_n such that $(x_{n+1}, y_{n+1}) = \varphi_4(x_n, y_n)$

5.2 Iterative Methods for Linear Systems

5.2.1 The Methods of Jacobi and Gauss Seidel

Review: These are iterative methods for solving a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is nonsingular and $\mathbf{b} \in \mathbb{R}^n$. We assume that the rows of \mathbf{A} are ordered so that the diagonal elements a_{ii} are nonzero for all i . Solving the i th equation of $\mathbf{A}\mathbf{x} = \mathbf{b}$ for the i th unknown $x(i)$, we obtain a **fixed-point form** of $\mathbf{A}\mathbf{x} = \mathbf{b}$

$$x(i) = \frac{1}{a_{ii}} \left(- \sum_{j=1}^{i-1} a_{ij}x(j) - \sum_{j=i+1}^n a_{ij}x(j) + b_i \right), \quad i = 1, 2, \dots, n. \quad (5.3)$$

Substituting \mathbf{x}_k into the right hand side of (5.3) we obtain a new approximation \mathbf{x}_{k+1} which is the **Jacobi method** in component form

$$(J) \quad \begin{aligned} x_{k+1}(i) &= \frac{1}{a_{ii}} \left(- \sum_{j=1}^{i-1} a_{ij}x_k(j) - \sum_{j=i+1}^n a_{ij}x_k(j) + b_i \right), \\ \text{for } i &= 0, 1, \dots, n, k = 0, 1, 2, \dots \end{aligned} \quad (5.4)$$

Carl Gustav Jacob Jacobi (1804–1851) was a German mathematician, who worked on elliptic functions, differential equations and number theory. He was professor of mathematics at Königsberg University from 1829.



The method of **Gauss-Seidel (GS)** is a modification of Jacobi's method, where we use the new $x_{k+1}(i)$ immediately after it has been computed.

$$(GS) \quad \left| \begin{array}{l} x_{k+1}(i) = \frac{1}{a_{ii}} \left(- \sum_{j=1}^{i-1} a_{ij} x_{k+1}(j) - \sum_{j=i+1}^n a_{ij} x_k(j) + b_i \right), \\ \text{for } i = 0, 1, \dots, n, k = 0, 1, 2, \dots \end{array} \right. \quad (5.5)$$

Johann Carl Friedrich Gauss (1777–1855) was a German mathematician and physical scientist who worked on many fields, including number theory, statistics, analysis, differential geometry, geodesy, geophysics, electrostatics, astronomy and optics. Gauss is known as one of history's most influential mathematicians.



Philipp Ludwig von Seidel (1821–1896) was a German mathematician, optical physicist and astronomer. He discovered the analytic concept of uniform convergence, while analyzing an incorrect proof of Cauchy and decomposed the first order monochromatic aberrations into five constituent aberrations. The Gauss-Seidel method is a useful numerical iterative method for solving linear systems.



Let $\mathbf{B} \in \mathbb{R}^{n \times n}$ and $\mathbf{c} \in \mathbb{R}^n$ and assume that the linear system $\mathbf{x} = \mathbf{Bx} + \mathbf{c}$ has a unique solution \mathbf{x}^* . Choose any $\mathbf{x}_0 \in \mathbb{R}^n$ and generate a sequence (\mathbf{x}_k) by the fixed-point iteration $\mathbf{x}_{k+1} = \mathbf{Bx}_k + \mathbf{c}$, $k = 0, 1, 2, \dots$. Then $\mathbf{x}_k \rightarrow \mathbf{x}^*$ if and only if all eigenvalues of \mathbf{B} are less than one in absolute value (modulus). A sufficient condition is that there exists a vector norm on \mathbb{R}^n and corresponding matrix norm such that $\|\mathbf{B}\| < 1$ (See Chap. 4 for different norms). \blacktriangleleft

Exercise 5.5. A first example

Let

$$\mathbf{A} := \begin{bmatrix} 5 & -2 & -2 \\ -2 & 5 & -3 \\ -2 & -3 & 5 \end{bmatrix}, \quad \mathbf{b} := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_0 = [0, 0, 0]^T. \quad (5.6)$$

1. Write a MATLAB script `Jacobitest` that computes approximations $\mathbf{y} \in \mathbb{R}^3$ to the exact solution \mathbf{z} of the system $\mathbf{Az} = \mathbf{b}$ using Jacobi's method. Iterate until $\|\mathbf{y} - \mathbf{z}\|_2 < 10^{-8}$. Return the number of iterations k_J used, or $k_J = 1001$ if the method did not converge in 1000 iterations.

```
>> Jacobitest
iterations :
309
```

2. Write a MATLAB script that does the same as in the last question, but uses Gauss-Seidel's method.

```
>> GStest
iterations :
155
```

Exercise 5.6. A second example

Consider the J- and GS methods applied to the linear system $\mathbf{A}\mathbf{x} = \mathbf{f}$, where for real parameters a, b

$$\mathbf{A} := \begin{bmatrix} 1 & 0 & a \\ 1 & 1 & 0 \\ b & 1 & 1 \end{bmatrix}, \quad \mathbf{f} := \begin{bmatrix} 1+a \\ 2 \\ 2+b \end{bmatrix}, \quad \mathbf{x}_0 := [0, 0, 0]^T. \quad (5.7)$$

1. Write the two methods in the form $\mathbf{x}_{k+1} = \mathbf{B}\mathbf{x}_k + \mathbf{c}$.
 2. Show that the characteristic polynomials $p(t) := \det(t\mathbf{I} - \mathbf{B})$ are given by
- $$p_J(t) = t^3 - (ab)t + a, \quad p_{GS}(t) = t^2(t - ab + a).$$
3. Show that the GS-method converges for any a if $b = 1$.
 4. Suppose $b = -1$. Show that the GS-method diverges if $|a| \geq 1/2$. It can be shown that $\rho(p_J(t)) \approx 0.65$ for $b = -1, a = 1/2$ so that the J-method converges.
- Comment:** For a proof see Stewart Venit, *The convergence of Jacobi and Gauss-Seidel iteration*, Mathematics Magazine vol 48, 1975, 163–167. ☺
5. Show that the J-method diverges for $a = 27/4, b = 1$.
 6. **Computation.** For $a = 2, b = 1$ do a few iterations with the J- and GS-method using MATLAB. Do the methods seem to converge? Same question with $a = 1/2, b = -1$.

Exercise 5.7. Convergence analysis

Consider the first example (5.6) given by

$$\mathbf{A} := \begin{bmatrix} 5 & -2 & -2 \\ -2 & 5 & -3 \\ -2 & -3 & 5 \end{bmatrix}, \quad \mathbf{b} := \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{x}_0 = [0, 0, 0]^T$$

and let \mathbf{z} be the exact solution of $\mathbf{A}\mathbf{z} = \mathbf{b}$.

1. Show that the J-method on this example is a fixed-point method where $\varphi_J(\mathbf{x}) := \mathbf{B}_J\mathbf{x} + \mathbf{c}_J$ and find \mathbf{B}_J and \mathbf{c}_J ▷ Math Hint⁴ ◁
2. Since \mathbf{B}_J is symmetric the spectral norm is equal to the **spectral radius** $\rho(\mathbf{B}_J)$, i.e., the largest eigenvalue of \mathbf{B}_J in absolute value. Compute the eigenvalues of \mathbf{B}_J and conclude on the convergence. Show that $\|\mathbf{x}_k - \mathbf{z}\|_2 \leq \sqrt{3}\rho(\mathbf{B}_J)^{k-1}$ for $k = 1, 2, 3, \dots$ ▷ Math Hint⁵ ◁
3. Compute the eigenvalues of \mathbf{B}_J using MATLAB. Compute k as the smallest integer larger than the solution r of $\sqrt{3}\tilde{\rho}(\mathbf{B}_J)^{r-1} = 10^{-8}$, where $\tilde{\rho}(\mathbf{B}_J)$ is the computed value of $\rho(\mathbf{B}_J)$. Compare with the number found in Exercise 5.5.

⁴ Answer:

$$\mathbf{B}_J = \frac{1}{5} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}, \quad \mathbf{c}_J := \frac{1}{5} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

⁵ Subtract $\mathbf{z} = \mathbf{B}_J\mathbf{z} + \mathbf{c}_J$ from $\mathbf{x}_{k+1} = \mathbf{B}_J\mathbf{x}_k + \mathbf{c}_J$.

4. Find \mathbf{B}_{GS} and \mathbf{c}_{GS} for the GS-method $\mathbf{x}_{k+1} = \mathbf{B}_{GS}\mathbf{x}_k + \mathbf{c}_{GS}$.
5. Show that the GS-method converges for the problem (5.6). ▷ Math Hint⁶ ◁
6. Show that $\|\mathbf{x}_k - \mathbf{z}\|_\infty \leq M^k$, $k = 1, 2, \dots$, where $M := \|\mathbf{B}_{GS}\|_\infty$.

Exercise 5.8. Convergence Analysis in the general case

We consider the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is nonsingular and $\mathbf{b} \in \mathbb{R}^n$. We assume that the rows of \mathbf{A} are ordered so that the diagonal elements a_{ii} are nonzero for all i . We derive three matrices from \mathbf{A} : the diagonal matrix $\mathbf{D} \in \mathbb{R}^{n \times n}$ with diagonal elements $d_{ii} = a_{ii}$, the upper triangular matrix $\mathbf{F} \in \mathbb{R}^{n \times n}$ with 0 on the diagonal and $f_{ij} = -a_{ij}$ for $j > i$, and the lower triangular matrix $\mathbf{E} \in \mathbb{R}^{n \times n}$ with 0 on the diagonal and $e_{ij} = -a_{ij}$ for $j < i$. Thus $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$.

1. Show that there exists $\mathbf{B}_J \in \mathbb{R}^{n \times n}$ and $\mathbf{c}_J \in \mathbb{R}^n$ such that the algorithms (J) in (5.4) can be written $\mathbf{x}_{k+1} = \mathbf{B}_J\mathbf{x}_k + \mathbf{c}_J$. ▷ Math Hint⁷ ◁
2. Same question with $\mathbf{B}_{GS} \in \mathbb{R}^{n \times n}$ and $\mathbf{c}_{GS} \in \mathbb{R}^n$ for (GS) in (5.5).
3. Give conditions on the spectral radius to ensure the convergence for each method.

5.2.2 Gradient Methods

The following exercises were proposed by our colleague Mounir Haddou.

Exercise 5.9. Testing gradients methods

Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a nonsingular matrix and $\mathbf{b} \in \mathbb{R}^n$. Again we deal with the solution of the linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$ by iterative methods and we propose testing algorithms for three gradient methods. We let $(\mathbf{u}, \mathbf{v}) := \mathbf{u}^T \mathbf{v}$ be the usual inner product on \mathbb{R}^n .

- Gradient Method with Constant parameter

$$(GMC) \quad \boxed{\begin{array}{l} \mathbf{x}_0 \in \mathbb{R}^n, \rho > 0, \\ \text{for } k = 0, 1, 2, \dots, \\ \mathbf{d}_k := \mathbf{b} - \mathbf{A}\mathbf{x}_k, \mathbf{x}_{k+1} := \mathbf{x}_k + \rho \mathbf{d}_k \end{array}}$$

- Gradient Method with Dynamic parameter (Steepest Descent)

$$(GMD) \quad \boxed{\begin{array}{l} \mathbf{x}_0 \in \mathbb{R}^n, \\ \text{for } k = 0, 1, 2, \dots \\ \mathbf{d}_k := \mathbf{b} - \mathbf{A}\mathbf{x}_k, \rho_k := \frac{(\mathbf{d}_k, \mathbf{d}_k)}{(\mathbf{A}\mathbf{d}_k, \mathbf{d}_k)}, \\ \mathbf{x}_{k+1} := \mathbf{x}_k + \rho_k \mathbf{d}_k, \end{array}}$$

⁶ Compute $\|\mathbf{B}_{GS}\|_\infty$.

⁷ Use $\mathbf{A} = \mathbf{D} - \mathbf{E} - \mathbf{F}$ to write \mathbf{B}_J and \mathbf{c}_J with the matrices $\mathbf{D}, \mathbf{E}, \mathbf{F}$ and the vector \mathbf{b} .

- Conjugate Gradient Method

$$(CGM) \quad \begin{cases} \mathbf{x}_0 \in \mathbb{R}^n, \mathbf{r}_0 = \mathbf{A}\mathbf{x}_0 - \mathbf{b}, \mathbf{d}_0 = -\mathbf{r}_0 \\ \textbf{for } k = 0, 1, 2, \dots \\ \rho_k := \frac{(\mathbf{r}_k, \mathbf{r}_k)}{(\mathbf{A}\mathbf{d}_k, \mathbf{d}_k)}, \mathbf{x}_{k+1} := \mathbf{x}_k + \rho_k \mathbf{d}_k \\ \mathbf{r}_{k+1} = \mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}, \\ \beta_k := \frac{(\mathbf{r}_{k+1}, \mathbf{r}_{k+1})}{(\mathbf{r}_k, \mathbf{r}_k)}, \\ \mathbf{d}_{k+1} := -\mathbf{r}_{k+1} + \beta_k \mathbf{d}_k \end{cases}$$

In any of these algorithm we can stop whenever $\|\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}\| \leq \text{precis}$ (in this case we consider that the algorithm has converged and the last computed \mathbf{x}_{k+1} is the approximation of the solution) or k becomes greater than a given maxiter (in this case we consider that the algorithm diverges).

1. **Program:** Write a MATLAB function file `[A, b] = construcmat(n)` for

$$\mathbf{A} = \begin{bmatrix} 4 & -2 & 0 & \dots & 0 & -1 \\ -2 & 4 & -2 & \ddots & & 0 \\ 0 & -2 & 4 & -2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & -2 & 4 & -2 & \\ -1 & 0 & \dots & 0 & -2 & 4 \end{bmatrix} \in \mathbb{R}^{n \times n}, \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ \vdots \\ 1 \\ 1 \end{bmatrix} \in \mathbb{R}^n.$$

▷ MATLAB Hint⁸▷ Test with $n = 5$.

```
>> [A, b] = construcmat(5)
A =
    4     -2      0      0     -1
   -2      4     -2      0      0
    0     -2      4     -2      0
    0      0     -2      4     -2
   -1      0      0     -2      4
b =
    1
    1
    1
    1
    1
```

2. Write three programs `GMC.m`, `GMD.m`, `CGM.m` for the algorithms. Test with the previous matrix \mathbf{A} and vector \mathbf{b} with $n = 10$, $\text{maxiter} = 100$, $\text{precis} = 10^{-10}$ (and $\rho = 0.1$ for GMC). Show the number of iterations and the error $\|\mathbf{x}_k - \mathbf{u}\|_2$ where \mathbf{u} is the solution of $\mathbf{A}\mathbf{u} = \mathbf{b}$ given by MATLAB.

⁸ Use `diag`, `ones`.

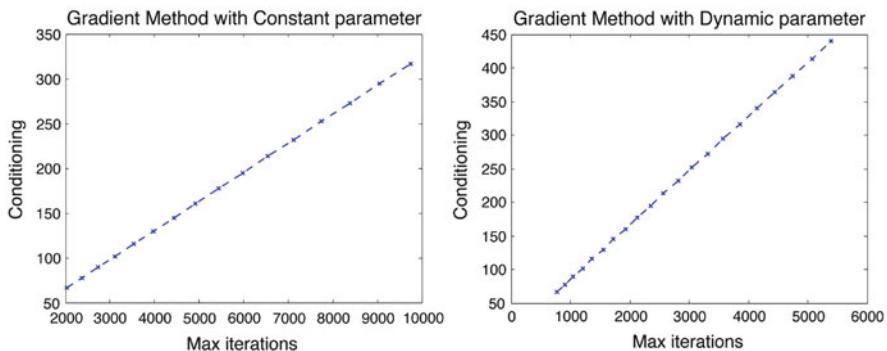


Fig. 5.3 GMC and GMD for dimension $n = 10, 11, \dots, 30$

```
>> GMC
error =
8.2840e-10
number of iterations:
iter =
2033
```

```
>> GMD
error =
5.6671e-10
number of iterations:
iter =
769
```

```
>> CGM
error =
2.4726e-14
number of iterations:
iter =
6
```

3. In this question we want to check numerically that the number of iterations to reach a given precision is proportional to $\text{cond}_2(\mathbf{A})$ for GMC and GMD and to $\sqrt{\text{cond}_2(\mathbf{A})}$ for CGM. \triangleright Math Hint⁹ \triangleleft Write the programs corresponding to the three algorithms, GMCvarp.m... Test from an array of dimensions $arrn = 10 : 30$. Give the corresponding graphs. See Figs. 5.3 and 5.4.

Exercise 5.10. Preconditioning the system

This exercise follows the previous one and uses some of its programs. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the tridiagonal symmetric matrix with elements $a_{ii} := 3i^2$ for $i = 1, \dots, p$

⁹ See Chap. 4 for the definition of conditioning.

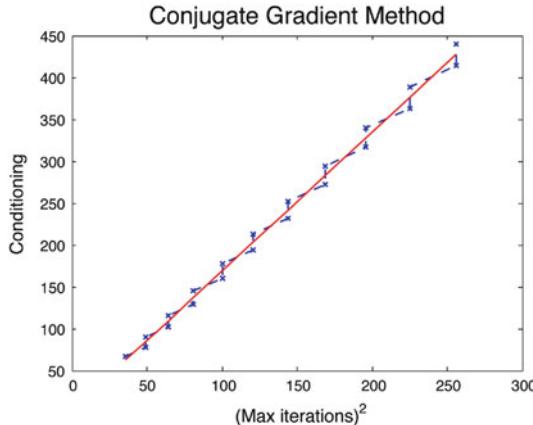


Fig. 5.4 CGM for dimension $n = 10, 11, \dots, 30$

and $a_{i,i+1} := -1$ for $i = 1, \dots, n-1$ and let $\mathbf{b} = [1, \dots, 1]^T \in \mathbb{R}^n$. Let \mathbf{u} be the solution of $\mathbf{A}\mathbf{u} = \mathbf{b}$.

1. Write a function `newconstructmat1.m` for the construction of \mathbf{A} and \mathbf{b} .

```
>> [A, ~] = newconstructmat1(5)
A =
    3     -1      0      0      0
   -1     12     -1      0      0
    0     -1     27     -1      0
    0      0     -1     48     -1
    0      0      0     -1     75
```

2. For $n = 1000$, compute numerically the number of iterations to obtain $\|\mathbf{Ax}_k - \mathbf{b}\|_2 < \eta := 10^{-10}$ for the Conjugate Gradient Method (CGM). The corresponding program will be labelled `CGMnew1.m`.

```
>> CGMnew1
error =
1.3460e-15
number of iterations:
iter =
1519
```

3. We introduce the diagonal matrix $\mathbf{C} \in \mathbb{R}^{n \times n}$ such that $c_{ii} = 1/i$ and we define $\tilde{\mathbf{A}} := \mathbf{CAC}$ and $\tilde{\mathbf{b}} := \mathbf{Cb}$. Let $\tilde{\mathbf{u}}$ be the solution of $\tilde{\mathbf{A}}\tilde{\mathbf{u}} = \tilde{\mathbf{b}}$. Thus the solution is $\mathbf{u} = \mathbf{C}\tilde{\mathbf{u}}$. Write a function `newconstructmat2.m` for the construction of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{b}}$.
4. For $n = 1000$, compute numerically the number of iterations to obtain $\|\tilde{\mathbf{A}}\mathbf{x}_k - \tilde{\mathbf{b}}\|_2 < \eta := 10^{-10}$ for CGM. The corresponding program will be labelled `CGMnew2.m`.

```
>> CGMnew2
error =
    1.0720e-12
number of iterations:
iter =
    8
```

5.3 Subdivision Schemes

We define $\begin{cases} \ell(\mathbb{Z}, \mathbb{R}^d) := \{\mathbf{f} : \mathbb{Z} \rightarrow \mathbb{R}^d\}, \\ \ell^\infty(\mathbb{Z}, \mathbb{R}^d) := \{\mathbf{f} : \mathbb{Z} \rightarrow \mathbb{R}^d : \|\mathbf{f}\|_\infty := \sup_{\alpha \in \mathbb{Z}} \|f(\alpha)\| < +\infty\} \end{cases}$
and for $\mathbf{f} \in \ell(\mathbb{Z}, \mathbb{R}^d)$, $\Delta f(\alpha) := f(\alpha + 1) - f(\alpha)$.

In the theoretical part of the following exercise we will assume that $d = 1$ and $\ell(\mathbb{Z}, \mathbb{R}^d)$, (resp. $\ell^\infty(\mathbb{Z}, \mathbb{R}^d)$) will be written $\ell(\mathbb{Z})$, (respec. $\ell^\infty(\mathbb{Z})$). The results for $d = 1$ can be extended to any dimension, component by component.

Exercise 5.11. The 4-point scheme

The algorithm proposed in this exercise was simultaneously created by Deslauriers, Dubuc and Dyn, Levin, Gregory in 1986–1987. For this too recent application created by our colleagues, we have not provided biographical details. The goal is the interpolation of data $\mathbf{f}_0 \in \ell(\mathbb{Z})$ by a continuous function $\varphi(t)$, $t \in \mathbb{R}$ depending on a parameter w .

For $0 < w < 1/2$, we start with $\mathbf{f}_0 \in \ell(\mathbb{Z})$ and generate the sequence $\mathbf{f}_n \in \ell(\mathbb{Z})$ recursively. From $\mathbf{f}_{n-1} \in \ell(\mathbb{Z})$, \mathbf{f}_n is defined by

$$\begin{cases} f_n(2\alpha) := f_{n-1}(\alpha) \\ f_n(2\alpha + 1) := \frac{1}{2}[f_{n-1}(\alpha) + f_{n-1}(\alpha + 1)] - w[f_{n-1}(\alpha - 1) + f_{n-1}(\alpha + 2)] \end{cases} \quad \alpha \in \mathbb{Z}. \quad (5.8)$$

1. Reproduction of polynomials: Show that if there exists $p \in \mathbb{P}_1$ such that $f_0(\alpha) = p(\alpha)$ for any $\alpha \in \mathbb{Z}$, then $f_n(\alpha) = p(\alpha 2^{-n})$. Show that this property can be extended to any $p \in \mathbb{P}_3$ whenever $w = 1/16$.
2. Show that for $n \geq 1$ and $\alpha \in \mathbb{Z}$,

$$\begin{aligned} & f_n(2\alpha + 1) - f_n(2\alpha) \\ &= \frac{1}{2}[f_{n-1}(\alpha + 1) - f_{n-1}(\alpha)] \\ &+ w[f_{n-1}(\alpha) - f_{n-1}(\alpha - 1)] + w[f_{n-1}(\alpha + 1) - f_{n-1}(\alpha + 2)], \end{aligned} \quad (5.9)$$

$$\begin{aligned} & f_n(2\alpha + 1) - f_n(2\alpha + 2) \\ &= \frac{1}{2}[f_{n-1}(\alpha) - f_{n-1}(\alpha + 1)] \\ &+ w[f_{n-1}(\alpha) + f_{n-1}(\alpha + 1)] + w[f_{n-1}(\alpha + 1) - f_{n-1}(\alpha + 2)]. \end{aligned} \quad (5.10)$$

3. In the following, we will assume that $\mathbf{f}_0 \in \ell(\mathbb{Z})$ and $\Delta \mathbf{f}_0 \in \ell^\infty(\mathbb{Z})$. Let $M_0 := \|\Delta \mathbf{f}_0\|_\infty$. Prove that $M_n := \|\Delta \mathbf{f}_n\|_\infty$ satisfies $M_n \leq \theta^n M_0$ where $\theta = 2|w| + 1/2$. \triangleright Math Hint¹⁰ \triangleleft
4. Let φ_n be the continuous piecewise linear function such that $\varphi_n(\alpha 2^{-n}) := f_n(\alpha)$ for $\alpha \in \mathbb{Z}$. Prove that for $t \in [\alpha 2^{-n}, (\alpha + 1)2^{-n}]$,

$$\begin{aligned} & |\varphi_{n+1}(t) - \varphi_n(t)| \\ & \leq w|f_n(\alpha) - f_n(\alpha - 1)| + w|f_n(\alpha + 1) - f_n(\alpha - 2)|. \end{aligned} \quad (5.11)$$

\triangleright Math Hint¹¹ \triangleleft

5. Deduce that for $|w| < 1/4$, there exists $\varphi \in C(\mathbb{R})$ such that $\varphi(\alpha/2^n) = f_n(\alpha)$ for $\alpha \in \mathbb{Z}$. \triangleright Math Hint¹² \triangleleft

\odot Comment: In the article: A 4-point interpolatory subdivision scheme for curve design by N. Dyn, D. Levin, J. Gregory, published in CAGD, it is also proved that φ is Hölder continuous, moreover for $0 < w < 1/8$, $\varphi \in C^1(\mathbb{R}^d)$.



6. **Programs:** From now on, we suppose that $\mathbf{f}_0 \in \ell(\mathbb{Z}, \mathbb{R}^2)$ and is periodic of period $T \in \mathbb{N}$, i.e., $\mathbf{f}_0(\alpha + T) = \mathbf{f}_0(\alpha)$ for all $\alpha \in \mathbb{Z}$. We define $\mathbf{f}0 := [f_0(1), \dots, f_0(T)] \in \mathbb{R}^{2 \times T}$.

- a. Show that the corresponding \mathbf{f}_n is also periodic with period $2^n T$.

We now suppose that $T \geq 3$.

- b. Write a MATLAB function `pt4iterat.m` that given a matrix $\mathbf{f}0 \in \mathbb{R}^{2 \times T}$ and w computes the corresponding $\mathbf{f}1 = [f_1(1), \dots, f_1(2T)] \in \mathbb{R}^{2 \times 2T}$. Test

```
>> f0=[0,2,1;0,0,1],w=1/16;f1=pt4iterat(f0,w)
f0 =
    0      2      1
    0      0      1
f1 =
    0.3125      0      1.0000      2.0000      1.6875      1.0000
    0.5625      0     -0.1250          0      0.5625      1.0000
```

- c. Write a function `pt4niterat.m` that with inputs $\mathbf{f}0 \in \mathbb{R}^{2 \times T}$, w , n computes the n -th iterate $\mathbf{f}n \in \mathbb{R}^{2 \times 2^{nT}}$. Test with the previous data and $n = 3$.

```
>> f0=[0,2,1;0,0,1],w=1/16;f=pt4niterat(f0,w,3)
f0 =
    0      2      1
    0      0      1
f3 =
    Columns 1 through 6
    0.8135      0.6328      0.4661      0.3125      0.1648      0.0508
    0.9492      0.8438      0.7122      0.5625      0.4109      0.2617
```

¹⁰ Prove that $M_n \leq \theta M_{n-1}$.

¹¹ The maximum value of $|\varphi_{n+1}(t) - \varphi_n(t)|$ is at the midpoint.

¹² $\varphi_n = \varphi_0 + \sum_{k=0}^{n-1} (\varphi_{k+1} - \varphi_k)$.

Columns 7 through 12	-0.0171	0	0.1694	0.4180	0.6987	1.0000
	0.1187	0	-0.0679	-0.1055	-0.1230	-0.1250
Columns 13 through 18	1.3013	1.5820	1.8306	2.0000	2.0171	1.9492
	-0.1230	-0.1055	-0.0679	0	0.1187	0.2617
Columns 19 through 24	1.8352	1.6875	1.5339	1.3672	1.1865	1.0000
	0.4109	0.5625	0.7122	0.8438	0.9492	1.0000

- d. Write a program calling the previous function and plotting the piecewise linear graphs of $[f_0, f_0(1)]$ and $[f_n, f_n(1)]$, where we add the first point to show the periodicity. Save as `pt4niterat.m`. Same test.
- e. Modify the previous program to do DAQ (Data AcQuisition) of f_0 with the mouse, cursors or buttons (see end of Exercise 7.1) to enter the number of iterations n and w . Save as `pt4getandplot.m`.

5.4 A Nonlinear Pendulum

Exercise 5.12. Pendulum oscillations

The following problem describes the oscillations of a pendulum subject to an external force f with given positions at times a and b :

$$(P) \begin{cases} u''(t) + \sin(u(t)) = f(t) \text{ for } t \in [a, b] \\ u(a) = \alpha, u(b) = \beta \end{cases}$$

The approximate problem is defined using finite differences. For a positive integer n and $h := (b - a)/(n + 1)$, we define a uniform partition on the interval $[a, b]$: $\mathbf{t} := [t_1, t_2, \dots, t_{n+2}]^T$ with $t_i := a + (i - 1)h$, for $i = 1, \dots, n + 2$. For $i = 2, \dots, n + 1$, $u''(t_i)$ is approximated by $\frac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2}$ (see (13.4)).

1. Show that the approximate problem can be written:

$$(P_h) : \mathbf{A}\mathbf{v} = h^2(\mathbf{f} - \sin \mathbf{v}) - \mathbf{b}$$

where

$$\mathbf{A} := \begin{bmatrix} -2 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & -2 \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad \mathbf{f} := \begin{bmatrix} f(t_2) \\ \vdots \\ f(t_k) \\ \vdots \\ f(t_{n+1}) \end{bmatrix} \in \mathbb{R}^n, \quad \mathbf{b} := \begin{bmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix} \in \mathbb{R}^n$$

The unknown $\mathbf{v} := [v_2, \dots, v_{n+1}]^T$ is the approximation of $\mathbf{u} := [u(t_2), \dots, u(t_{n+1})]^T$ while $\bar{\mathbf{v}} := \begin{bmatrix} \alpha \\ \mathbf{v} \\ \beta \end{bmatrix}$ approximates $\bar{\mathbf{u}} := \begin{bmatrix} \alpha \\ \mathbf{u} \\ \beta \end{bmatrix}$, the solution of (P) at the discrete points.

The equation in (P_h) is nonlinear. To solve it, we use an iterative method starting with $\mathbf{v}^0 := [0, \dots, 0]^T \in \mathbb{R}^n$ and then for $k = 0, 1, 2, \dots$ compute \mathbf{v}^{k+1} by solving the linear system

$$\mathbf{A}\mathbf{v}^{k+1} = h^2(\mathbf{f} - \sin \mathbf{v}^k) - \mathbf{b}. \quad (5.12)$$

We accept without proof that the iterative method converges and that after a finite number of iterations, the approximate solution is sufficiently accurate.

2. **Programming :** Construct the vector \mathbf{t} . Test with $n = 3, a = 0$ and $b = 1$. Save as `pendull.m`.

```
pendull
t =
    0
0.2500
0.5000
0.7500
1.0000
```

3. For $\alpha := 1$ and $\beta := -1$, if $f(t) = f_1(t) := \sin(\sin(\pi(t+1/2))) - \pi^2 \sin(\pi(t+1/2))$, then the solution of (P) is $u(t) = u_1(t) := \sin(\pi(t+1/2))$. Write two function files `f1.m` and `u1.m`.

```
f1 ([0.1 1])
ans =
-8.5725      9.0281
u1 ([0.1 1])
ans =
0.9511     -1.0000
```

4. Extend the first program with the construction of \mathbf{f} , \mathbf{b} and $\bar{\mathbf{u}}$. Save as `pendu12.m`. Test with $n = 3, a = 0, b = 1$. Display \mathbf{f} , \mathbf{b} and $\bar{\mathbf{u}}$.

```
pendu12
f =
-6.3292
-0.0000
6.3292
vb =
1
0
-1
bu =
1.0000
0.7071
0.0000
-0.7071
-1.0000
```

5. Extend the second program with the construction of the matrix \mathbf{A} . Save as pendul3.m. Test with the previous data. Display \mathbf{A} . ◇ MATLAB Hint¹³ ◇

```
pendul3
A =
-2      1      0
 1     -2      1
 0      1     -2
```

6. Apply the iterative method (5.12) p times. ◇ MATLAB Hint¹⁴ ◇ Display $\bar{\mathbf{v}} = \begin{bmatrix} \alpha \\ \mathbf{v}^{p+1} \\ \beta \end{bmatrix}$ and the error at each step. Save as pendul4.m. Test with $p = 6$ and the previous data. See Fig. 5.5.

```
pendul4
bv =
 1.0000
 0.7184
 0.0000
 -0.7184
 -1.0000
erriter =
 0.6978    0.0201    0.0005    0.0000    0.0000
 0.0000    0.0000
```

7. Extend the previous program with the graphs of the solutions of (P) and (P_h) and the computation of the error $\|\bar{\mathbf{u}} - \bar{\mathbf{v}}\|_\infty = \max_{1 \leq i \leq n+2} |u(t_i) - v_i|$. Save as pendul5.m. Test with the same data.

```
pendul5
error =
 0.0112
```

8. Write a program pendulorder.m to study the error when modifying n . Start with an array $\mathbf{n} = 20 : 10 : 200$ then plot $\log(error)$ as a function of $n + 1$. Compute a numerical estimation of the order of the method (Fig. 5.6). Test with the previous data and the number of iterations $p = 2n$. What seems to be the order of convergence?

9. The **damped pendulum equation** can be written:

$$(P_2) \begin{cases} u''(t) + u'(t) + \sin(u(t)) = f(t), t \in [a, b] \\ u(a) = \alpha, u(b) = \beta \end{cases}$$

¹³ Use the MATLAB functions `ones`, `diag`.

¹⁴ Starting with \mathbf{x}^0 , at each step compute \mathbf{x}^1 such that $\mathbf{Ax}^1 = h^2(f - \sin \mathbf{x}^0) - \mathbf{b}$ where \mathbf{x}^0 is the previous value and \mathbf{x}^1 the new one, compute the error $\|\mathbf{x}^1 - \mathbf{x}^0\|_\infty$ and save it in an array `erriter(k)` with $p + 1$ components. Update with $\mathbf{x}^0 = \mathbf{x}^1$.

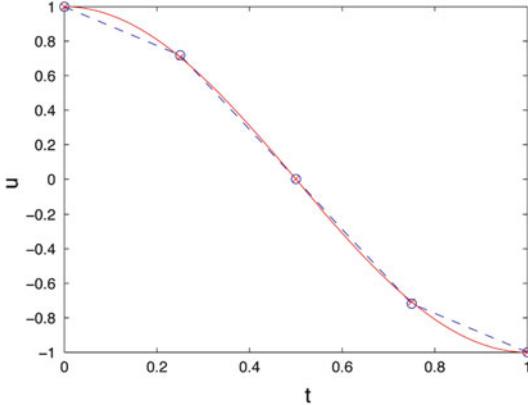


Fig. 5.5 Pendulum: exact and approximate solution for $n = 3$

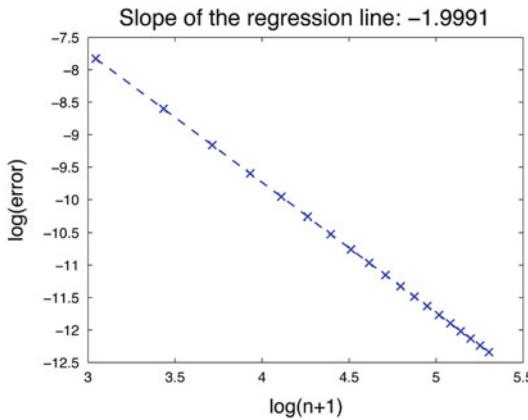


Fig. 5.6 Pendulum: Error between exact and approximate solutions

We choose $a := 0$, $b := 1$, $\alpha := 0$, $\beta := 0$ and $f(t) = f_2(t) := \sin(\cos(\pi(t + 1/2))) - \pi \sin(\pi(t + 1/2)) - \pi^2 \cos(\pi(t + 1/2))$. Thus the solution is $u(t) = u_2(t) := \cos(\pi(t + 1/2))$.

Now for $i = 2, \dots, n + 1$, $u'(t_i)$ is approximated by $\frac{u(t_{i+1}) - u(t_{i-1})}{2h}$, (see (13.3)). Show that the approximate problem can be written

$$(P'_h) : (\mathbf{A} + \frac{h}{2}\mathbf{C})\mathbf{v} = h^2(\mathbf{f} - \sin \mathbf{v}) - \frac{h}{2}\mathbf{d} - \mathbf{b}$$

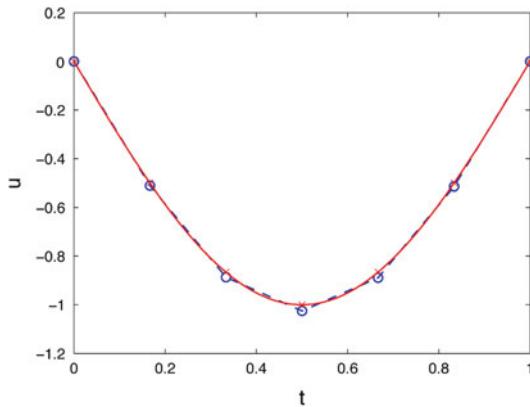


Fig. 5.7 Damp Pendulum

where

$$\mathbf{C} = \begin{bmatrix} 0 & 1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n} \text{ and } \mathbf{d} = \begin{bmatrix} -\alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{bmatrix} \in \mathbb{R}^n$$

Write a program `damppendu.m` that computes the solution \mathbf{v} of (P'_h) , then plots the corresponding $\bar{\mathbf{v}}$ and $\bar{\mathbf{u}}$. $a = 0, b = 1, \alpha = \beta = 0$. See Fig. 5.7.

```
>>damppendu
n =
      5
bv =
      0
-0.5112
-0.8867
-1.0252
-0.8890
-0.5140
      0
error =
    0.0252
```

10. Write a program `damppendulorder.m` to study the error when modifying n .

5.5 Solutions

5.5.1 Fixed Point Methods

Solution of Exercise 5.1

1. $\varphi_1(\sqrt{2}) = \frac{1}{2} \left(\sqrt{2} + \frac{2}{\sqrt{2}} \right) = \sqrt{2}$. If $\frac{1}{2} \left(x + \frac{2}{x} \right) = x$ then $x^2 + 2 = 2x^2$ or $x^2 = 2$. Thus $-\sqrt{2}$ is also a fixed-point.
2. squareroot2.m

```
function e=squareRoot2(x,n)
format short e;
s2=sqrt(2); e=zeros(n,1); e(1)=x-s2;
for k=1:n
    x=(x+2/x)/2; e(k+1)=x-s2;
end
```

Since $e_{k+1} \sim e_k^2$ for $k = 2, 3$ the convergence seems to be of order 2.

3. Since $\varphi'_1(x) = \frac{1}{2} \left(1 - \frac{2}{x^2} \right)$ it follows that $0 < \varphi'_1(x) < 1$ for $x \in I = (\sqrt{2}, +\infty)$. If $x \in I$ then $\varphi_1(x) > \varphi_1(\sqrt{2}) = \sqrt{2}$ since φ_1 is increasing on I . It follows that $\varphi_1(x) \in I$. But then the convergence result follows.
4. Suppose $0 < x_0 < \sqrt{2}$. Then $x_1 = \frac{x_0}{2} + \frac{1}{x_0} > \frac{1}{x_0} > \sqrt{2}$. Thus it is enough to consider the case $x_0 > \sqrt{2}$ and we already showed convergence for any such x_0 .
5. We find

$$x_{k+1} - \sqrt{2} = \frac{1}{2} \left(x_k + \frac{2}{x_k} \right) - \sqrt{2} = \frac{1}{2x_k} (x_k^2 + 2 - 2\sqrt{2}x_k) = \frac{1}{2x_k} (x_k - \sqrt{2})^2$$

and (5.2) follows. Since $x_k \rightarrow \sqrt{2}$ it follows from (5.2) that

$$\lim_{k \rightarrow \infty} \frac{e_{k+1}}{e_k^2} = \frac{1}{2\sqrt{2}}$$

and the order of convergence is $p = 2$.

6. The iteration function for Newton's method becomes

$$\varphi(x) := x - \frac{f(x)}{f'(x)} = x - \frac{x^2 - 2}{2x} = x - \frac{1}{2}x + \frac{1}{x} = \varphi_1(x).$$

Solution of Exercise 5.2

1. For any $x \in (0, 1]$, $\varphi_2(x) := \sin x^a < x^a \leq x$ and $\varphi_2(0) = 0$, thus the only fixed-point of φ_2 on $[0, 1]$ is $x^* = 0$ and also $\varphi_2([0, 1]) \subset [0, 1]$.

2. $0 \leq x_{n+1} = \sin x_n^a \leq x_n^a \leq x_n \leq 1$ and the sequence x_n is decreasing and have a lower bound. Thus it converges and the only possible limit is the fixed-point $x^* = 0$.
3. phi2.m

```
function v=phi2(u,a)
v=sin(u^a);
```

4. fixpoint1.m

```
a=3;
xinit=1;maxiter=20;precis=1e-10;
error=1;iter=1;u(1)=xinit;
while (error>precis)&(iter<maxiter)
    v=phi2(u(iter),a);
    error=abs(v-u(iter));
    iter=iter+1;
    u(iter)=v;
end
if iter==maxiter
    disp('No convergence')
else
    disp('lim = '), u(iter)
    disp('number of iterations: '), iter
end
```

5. fixpoint2.m

```
fixpoint1
plot(1:iter,u,'x-')
```

6. fixpoint3.m

```
fixpoint1
subplot(121), plot(1:iter,u,'x-')
title(['a = ',num2str(a)])
xlabel('iteration'), ylabel('x_n')
err=abs(u-u(iter));
subplot(122), plot(log(err(1:iter-2)),log(err(2:iter-1)),'o--')
c=polyfit(log(err(1:iter-2)),log(err(2:iter-1)),1);
order=c(1)
title(['Approx of the order: ',num2str(order)])
xlabel('log(e_n)'), ylabel('log(e_{n+1})')
```

The computations give a as the convergence order.

7. Since $\frac{e_{n+1}}{e_n^a} := \frac{x_{n+1} - 0}{(x_n - 0)^a} = \frac{\sin(x_n^a)}{x_n^a}$ and since the sequence x_n converges to 0, we deduce that $\lim_{n \rightarrow +\infty} \frac{e_{n+1}}{e_n^a} = 1$ to conclude that the order is a .

Solution of Exercise 5.3

1. For $x \in [0, 1]$, $x(1-x) \in [0, 1/4]$ and for $\lambda \in [0, 4]$, $\varphi_3([0, 1]) \subset [0, 1]$.

2. For $\lambda \neq 0$, $\varphi_3(x) = x \Leftrightarrow x = \lambda x(1-x)$. The solutions are $x = 0$ or $1 = \lambda(1-x)$ which gives $x = x^* := \frac{\lambda - 1}{\lambda}$.

Since 0 and x^* are also solutions of $\varphi_3 \circ \varphi_3(x) = x$, we obtain

$$\begin{aligned}\varphi_3 \circ \varphi_3(x) - x &= \lambda \varphi_3(x)(1 - \varphi_3(x)) - x \\ &= \lambda^2 x(1-x)(1 - \lambda x(1-x)) - x \\ &= x(\lambda^3 x^3 - 2\lambda^3 x^2 + \lambda^2(1-\lambda)x + \lambda^2 - 1) \\ &= x\left(x - \frac{\lambda - 1}{\lambda}\right)(\lambda^3 x^2 - \lambda^2(\lambda + 1)x + \lambda(\lambda + 1))\end{aligned}$$

Thus, for $\lambda \in [3, 4]$ there exist two other roots of $\varphi_3 \circ \varphi_3(x) = x$:

$$r^* := \frac{\lambda + 1 + \sqrt{(\lambda + 1)(\lambda - 3)}}{2\lambda} \text{ and } t^* := \frac{\lambda + 1 - \sqrt{(\lambda + 1)(\lambda - 3)}}{2\lambda}.$$

We notice that $\varphi_3(r^*) = t^*$ and $\varphi_3(t^*) = r^*$.

3. phi3.m

```
function v=phi3(u,lambda)
v=lambda*u*(1-u);
```

4. newfixpoint.m

```
xinit=0.1; maxiter=500; precis=1e-10;
lambda=1.5;
error=1; iter=1; u(1)=xinit;
while (error>precis)&(iter<maxiter)
    v=phi3(u(iter),lambda);
    error=abs(v-u(iter));
    iter=iter+1;
    u(iter)=v;
end
if iter==maxiter
    disp('No convergence')
    plot(1:iter,u,'x-')
else
    disp('lim = '), u(iter)
    disp('number of iterations: '), iter
    subplot(121), plot(1:iter,u,'x-')
    title(['lambda = ',num2str(lambda)])
    err=abs(u-u(iter));
    subplot(122), plot(log(err(1:iter-2)),log(err(2:iter-1)))
    a=polyfit(log(err(1:iter-2)),log(err(2:iter-1)),1);
    order=a(1)
    title(['Approx of the order: ',num2str(order)])
end
```

The convergence depends on the values of λ ,

$\lambda = 0.5$ then the sequence converges to 0 with order 1.

$\lambda = 1.5$ then the sequence converges to x^* with order 1.

$\lambda = 3.1$ then, for n large enough, the sequence is almost periodic and oscillates between the two values $r^* \approx 0.7646$ and $t^* \approx 0.5580$.

For larger values of λ the sequence is also almost periodic with periods 2, 4 etc...

Solution of Exercise 5.4

$$1. \quad r^* = \begin{bmatrix} b - 1 + \sqrt{(b-1)^2 + 4a} \\ \frac{2a}{b(b-1 + \sqrt{(b-1)^2 + 4a})} \end{bmatrix} \text{ and } t^* = \begin{bmatrix} b - 1 - \sqrt{(b-1)^2 + 4a} \\ \frac{2a}{b(b-1 - \sqrt{(b-1)^2 + 4a})} \end{bmatrix}.$$

2. phi4.m

```
function v=phi3(u,a,b)
v=[1-a*u(1)^2+u(2);b*u(1)];
```

fixpointdim2.m

```
maxiter=100; precis=1e-10;
a=0.1;b=0.3;
uinit=[1;1]; error=1; iter=1;u=uinit;
while (error>precis)&(iter<maxiter)
    v=phi4(u(:,iter),a,b);
    error=norm(v-u(:,iter));
    iter=iter+1;
    u(:,iter)=v;
end
if iter==maxiter
    disp('No convergence')
else
    disp('lim = ')
    u(:,iter)
    disp('number of iterations: ')
    iter
end
subplot(121), plot(1:iter,u(1,:))
xlabel('n, iteration'), ylabel('x_n')
title(['a = ',num2str(a), ' b = ',num2str(b)])
subplot(122), plot(1:iter,u(2,:))
xlabel('n, iteration'), ylabel('y_n')
```

5.5.2 Iterative Methods for Linear Systems

Solution of Exercise 5.5

1. Jacobitest.m

```
x=[0,0,0]; f=[1,1,1]; y=x;
for k=1:1000
    y(1)=(2*x(2)+2*x(3)+1)/5;
    y(2)=(2*x(1)+3*x(3))/5;
    y(3)=(2*x(1)+3*x(2))/5;
    if norm(y-f)<10^(-8) disp('iterations : '),disp(k); return
    end
```

```

x=y;
end
k=1001;

```

2. GStest.m

```

x=[0 ,0 ,0]; f =[1 ,1 ,1];
for k=1:1000
    x(1)=(2*x(2)+2*x(3)+1)/5;
    x(2)=(2*x(1)+3*x(3))/5;
    x(3)=(2*x(1)+3*x(2))/5;
    if norm(x-f)<10^(-8) disp(k); return
end
k=1001;

```

Comment: We see that $k_J/k_{GS} \approx 2$. ☺

Solution of Exercise 5.6.

- For the J-method we find

$$\begin{aligned} x_{k+1}(1) &= -ax_k(3) + 1 + a, \\ x_{k+1}(2) &= -x_k(1) + 2, \\ x_{k+1}(3) &= -bx_k(1) - x_k(2) + 2 + b, \end{aligned}$$

giving

$$\mathbf{B}_J = \begin{bmatrix} 0 & 0 & -a \\ -1 & 0 & 0 \\ -b & -1 & 0 \end{bmatrix}, \quad \mathbf{c}_J = \begin{bmatrix} 1+a \\ 2 \\ 2+b \end{bmatrix}.$$

and for the GS-method we find

$$\begin{aligned} x_{k+1}(1) &= -ax_k(3) + 1 + a, \\ x_{k+1}(2) &= -x_{k+1}(1) + 2 = -(-ax_k(3) + 1 + a) + 1 + a \\ &= ax_k(3) + 1 - a, \\ x_{k+1}(3) &= -bx_{k+1}(1) - x_{k+1}(2) + 2 + b \\ &= -b(-ax_k(3) + 1 + a) - (ax_k(3) + 1 - a) + 2 + b \\ &= (ab - a)x_k(3) + 1 + a - ab, \end{aligned}$$

giving

$$\mathbf{B}_{GS} = \begin{bmatrix} 0 & 0 & -a \\ 0 & 0 & a \\ 0 & 0 & ab - a \end{bmatrix}, \quad \mathbf{c}_{GS} = \begin{bmatrix} 1+a \\ 1-a \\ 1+a-ab \end{bmatrix}.$$

2.

$$\det(t\mathbf{I} - \mathbf{B}_J) = \begin{vmatrix} t & 0 & a \\ 1 & t & 0 \\ b & 1 & t \end{vmatrix} = t \begin{vmatrix} t & 0 \\ 1 & t \end{vmatrix} + a \begin{vmatrix} 1 & t \\ b & 1 \end{vmatrix} = t^3 + a - abt = p_J(t).$$

$$\det(t\mathbf{I} - \mathbf{B}_{GS}) = \begin{vmatrix} t & 0 & a \\ 0 & t & -a \\ 0 & 0 & t - ab + a \end{vmatrix} = p_{GS}(t).$$

- 3. If $b = 1$ then $p_{GS}(t) = t^3$ for any value of a . Thus all eigenvalues are zero and the method converges.
- 4. If $b = -1$ then $p_{GS}(t) = t^2(t + 2a)$. Thus one eigenvalue has absolute value, $|2a| \geq 1$ and the method diverges.
- 5. If $b = 1$ and $a = 27/4$ then $p_J(t) = t^3 + 27/4t + 27/4 = (t - 3/2)^2(t + 3)$. Thus since all eigenvalues have absolute value greater than 1, the method diverges.
- 6.

```
disp('Jacobi');
a=2
b=1
a=2; b=1;
B=[0,0,-a; -1,0,0; -b,-1,0];
f=[1+a,2,2+b]';
X=zeros(3,6);
for k=1:5
    X(:,k+1)=B*X(:,k)+f;
end
disp(X)
disp('Jacobi');
a=1/2
b=-1
B=[0,0,-a; -1,0,0; -b,-1,0];
f=[1+a,2,2+b]';
X=zeros(3,50);
for k=1:50
    X(:,k+1)=B*X(:,k)+f;
end
disp(X(:,50))
```

```
>> Jacobi2
Jacobi

a =
2
b =
1

0      3      -3      7      -11      21
0      2      -1      5      -5      13
0      3      -2      7      -9      19
```

```
Jacobi
a =
    0.5000
b =
    -1
    0.9985
    0.9953
    1.009
```

It appears that Jacobi diverges for $a = 2$, $b = 1$ and converges for $a = 1/2$, $b = -1$.

```
disp('Gauss-Seidel');
a=2
b=1
B=[0,0,-a; -1,0,0; -b,-1,0];
f=[1+a,2,2+b]';
X=zeros(3,6);
for k=1:5
    X(1,k+1)=-a*X(3,k)+1+a;
    X(2,k+1)=-X(1,k+1)+2;
    X(3,k+1)=-b*X(1,k+1)-X(2,k+1)+2+b;
end
disp(X)
disp('Gauss-Seidel');
a=1/2
b=-1
B=[0,0,-a; -1,0,0; -b,-1,0];
f=[1+a,2,2+b]';
X=zeros(3,6);
for k=1:5
    X(1,k+1)=-a*X(3,k)+1+a;
    X(2,k+1)=-X(1,k+1)+2;
    X(3,k+1)=-b*X(1,k+1)-X(2,k+1)+2+b;
end
disp(X)
```

```
>> GS2
Gauss-Seidel

a =
    2
b =
    1

    0      3      1      1      1      1
    0     -1      1      1      1      1
    0      1      1      1      1      1
```

Gauss-Seidel
a =
0.5000
b =
-1
0 1.5000 0.5000 1.5000 0.5000 1.5000
0 0.5000 1.5000 0.5000 1.5000 0.5000
0 2.0000 0 2.0000 0 2.0000

It appears that Gauss-Seidel converges for $a = 2, b = 1$ and diverges for $a = 1/2, b = -1$.

Solution of Exercise 5.7

1.

$$\begin{aligned}x_{k+1}(1) &= \frac{2}{5}x_k(2) + \frac{2}{5}x_k(3) + \frac{1}{5}, \\x_{k+1}(2) &= \frac{2}{5}x_k(1) + \frac{3}{5}x_k(3), \\x_{k+1}(3) &= \frac{2}{5}x_k(1) + \frac{3}{5}x_k(2),\end{aligned}$$

giving

$$\mathbf{B}_J = \frac{1}{5} \begin{bmatrix} 0 & 2 & 2 \\ 2 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}, \quad \mathbf{c}_J := \frac{1}{5} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}.$$

2. To find the eigenvalues of \mathbf{B} we compute the determinant

$$\begin{aligned}|\mathbf{B}_J - t\mathbf{I}| &= \frac{1}{125} \begin{vmatrix} -5t & 2 & 2 \\ 2 & -5t & 3 \\ 2 & 3 & -5t \end{vmatrix} \\&= \frac{1}{125} [-5t(25t^2 - 9) + 2(6 + 10t) + 2(6 + 10t)] \\&= \frac{1}{125}(5t + 3)[-5t(5t - 3) + 8] \\&= \frac{1}{125}(5t + 3)(-25t^2 + 15t + 8).\end{aligned}$$

The roots of the determinant are $-\frac{3}{5}, \frac{3 + \sqrt{41}}{10} \approx 0.9403$ and $\frac{3 - \sqrt{41}}{10} \approx -0.3403$. All the eigenvalues have absolute value less than 1 and the Jacobi method converges.

With the difference of $\mathbf{x}_{k+1} = \mathbf{B}_J \mathbf{x}_k - \mathbf{c}_J$ and $\mathbf{z} = \mathbf{B}_J \mathbf{z} - \mathbf{c}_J$, we deduce that $\mathbf{x}_{k+1} - \mathbf{z} = \mathbf{B}_J(\mathbf{x}_k - \mathbf{z})$ thus $\|\mathbf{x}_{k+1} - \mathbf{z}\|_2 \leq \|\mathbf{B}_J\|_2 \|\mathbf{x}_k - \mathbf{z}\|_2$ and by induction $\|\mathbf{x}_k - \mathbf{z}\|_2 \leq \|\mathbf{B}_J\|_2^k \|\mathbf{z}\|_2$ since $\mathbf{x}_0 = \mathbf{0}$.

Since the solution of the system is $\mathbf{z} = [1, 1, 1]^T$ with $\|\mathbf{z}\|_2 = \sqrt{3}$, we obtain the result.

3.

```
>> l=eig([0,2,2; 2,0,3; 2,3,0]/5)
l =
    -0.6000
   -0.3403
    0.9403
>> k=round(-(8+log10(sqrt(3)))/log10(l(3))+1)
k =
    309
```

Comment: This indicates that the rate of convergence is closely related to the size of the spectral radius of the iteration matrix \mathbf{B} . ☺

4.

$$\begin{aligned}x_{k+1}(1) &= \frac{2}{5}x_k(2) + \frac{2}{5}x_k(3) + \frac{1}{5}, \\x_{k+1}(2) &= \frac{2}{5}x_{k+1}(1) + \frac{3}{5}x_k(3) = \frac{4}{25}x_k(2) + \frac{4}{25}x_k(3) + \frac{2}{25} + \frac{3}{5}x_k(3), \\&= \frac{4}{25}x_k(2) + \frac{19}{25}x_k(3) + \frac{2}{25} \\x_{k+1}(3) &= \frac{2}{5}x_{k+1}(1) + \frac{3}{5}x_{k+1}(2) \\&= \frac{4}{25}x_k(2) + \frac{4}{25}x_k(3) + \frac{2}{25} + \frac{12}{125}x_k(2) + \frac{57}{125}x_k(3) + \frac{6}{125} \\&= \frac{32}{125}x_k(2) + \frac{77}{125}x_k(3) + \frac{16}{125}\end{aligned}$$

giving

$$\mathbf{b}_{GS} = \frac{1}{125} \begin{bmatrix} 0 & 50 & 50 \\ 0 & 20 & 95 \\ 0 & 32 & 77 \end{bmatrix} \quad \mathbf{c}_{GS} = \frac{1}{125} \begin{bmatrix} 25 \\ 10 \\ 16 \end{bmatrix}.$$

5. The method converges since $\|\mathbf{B}_{GS}\|_\infty = \max\{50+50, 20+95, 32+77\}/125 = 115/125 < 1$.
6. For $k = 0, 1, 2, \dots$, we find $\mathbf{x}_{k+1} - \mathbf{z} = \mathbf{B}_{GS}(\mathbf{x}_k - \mathbf{z})$ thus $\mathbf{x}_k - \mathbf{z} = \mathbf{B}_{GS}^k(\mathbf{x}_0 - \mathbf{z})$ and we obtain the bound since $\|\mathbf{z}\|_\infty = 1$.

Solution of Exercise 5.8

1. Equation (5.4) can be written $\mathbf{D}\mathbf{x}_{k+1} = (\mathbf{E} + \mathbf{F})\mathbf{x}_k + \mathbf{b}$ thus $\mathbf{x}_{k+1} = \mathbf{B}_J\mathbf{x}_k + \mathbf{c}_J$ with $\mathbf{B}_J = \mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})$ and $\mathbf{c}_J = \mathbf{D}^{-1}\mathbf{b}$. We notice that the diagonal matrix \mathbf{D} is nonsingular since the diagonal elements satisfy $d_{ii} = a_{ii} \neq 0$.
2. Equation (5.5) can be written $\mathbf{D}\mathbf{x}_{k+1} = \mathbf{E}\mathbf{x}_{k+1} + \mathbf{F}\mathbf{x}_k + \mathbf{b}$ or $(\mathbf{D} - \mathbf{E})\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{b}$ and thus $\mathbf{x}_{k+1} = \mathbf{B}_{GS}\mathbf{x}_k + \mathbf{c}_{GS}$ with $\mathbf{B}_{GS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{F}$ and $\mathbf{c}_{GS} = (\mathbf{D} - \mathbf{E})^{-1}\mathbf{b}$. We notice that the lower triangular matrix $\mathbf{D} - \mathbf{E}$ is nonsingular since the diagonal elements are nonzero.

3. The Jacobi method converges as soon as $\rho(\mathbf{D}^{-1}(\mathbf{E} + \mathbf{F})) < 1$ and Gauss-Seidel method converges as soon as $\rho((\mathbf{D} + \mathbf{E})^{-1}\mathbf{F}) < 1$ where $\rho(\cdot)$ is the spectral radius (See Chap. 3).

Solution of Exercise 5.9

1. constructmat.m

```
function [A,b]=constructmat(n)
A=2*eye(n)-2*diag(ones(n-1,1),1);
A=A+A';
A(1,n)=-1;
A(n,1)=-1;
b=ones(n,1);
```

2. GMC.m

```
maxiter=10000;
precis=1e-10; error=1;
iter=1;
n=10;
[A,b]=constructmat(n);
u=zeros(n,1);
rho=0.1;
while (error>precis)&(iter<maxiter)
    d=b-A*u; error=norm(d,2);
    v=u+rho*d;
    iter=iter+1; u=v;
end
if iter==maxiter
    disp('No convergence')
else
    error=norm(u-A\b,2)
    disp('number of iterations: ')
    iter
end
```

GMD.m is similar to GMC except

```
while (error>precis)&(iter<maxiter)
    d=b-A*u; error=norm(d,2);
    rho=(d'*d)/(d'*A*d);
    v=u+rho*d;
    iter=iter+1; u=v;
end
```

CGM.m

```
maxiter=100; precis=1e-10;
iter=1; error=1;
n=10;
[A,b]=constructmat(n);
u=zeros(n,1);
r=A*u-b; d=-r;
```

```

while (error>precis)&(iter<maxiter)
    rho=(r'*r)/(d'*A*d); v=u+rho*d; u=v;
    rnew=A*u-b; beta=(rnew'*rnew)/(r'*r);
    d=rnew+beta*d;
    r=rnew; iter=iter+1; error=norm(r,2);
end
if iter==maxiter
    disp('No convergence')
else
    error=norm(u-A\b,2)
    disp('number of iterations: '), iter
end

```

3. GMCvarn.m

```

maxiter=10000; precis=1e-10;
iter=1; error=1;
arrn=10:30
for j=1:length(arrn)
    n=arrn(j)
    error=1; iter=1;
    [A,b]=construcmat(n); u=zeros(n,1);
    rho=0.1;
    while (error>precis)&(iter<maxiter)
        d=b-A*u; error=norm(d,2);
        v=u+rho*d;
        iter=iter+1; u=v;
    end
    if iter==maxiter
        disp('No convergence')
    else
        nmaxiter(j)=iter
        condit(j)=cond(A,2);
    end
end
plot(nmaxiter,condit,'x—')
xlabel('Max iterations'), ylabel('Conditioning')
title('Gradient Method with Constant parameter')

```

GMDvarn.m and CGMvarn.m are similar to GMCvarn except inside the loop on the dimension of the matrix. We only give the end of CGMvarn.m where the conditioning is a function of $nmaxiter^2$.

```

a=polyfit(nmaxiter.^2,condit,1)
plot(nmaxiter.^2,condit,'x—',nmaxiter.^2,a(1)*nmaxiter.^2+a(2))
xlabel('(Max iterations)^2'), ylabel('Conditioning')
title('Conjugate Gradient Method')

```

Comment: With the MATLAB function polyfit we compute the coefficient of the regression line, see Chap. 10. ☺

Solution of Exercise 5.10

1. newconstructmat1.m

```
function [A,b]=newconstructmat1(n)
A=3*diag([1:n].^2)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
b=ones(n,1);
```

2. CGMnew1.m is similar to CGM.m except for the construction of the matrix A and the vector b where we use newconstructmat1.m instead of constructmat.m.

3. newconstructmat2.m

```
function [A,b]=newconstructmat2(n)
A=3*diag([1:n].^2)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
b=ones(n,1);
C=diag(1./[1:n]);
A=C*A*C;
b=C*b;
```

4. CGMnew2.m is similar to CGM.m with newconstructmat2.m.

⊕**Comment:** With the command $\text{cond}(\cdot, 2)$, we can compute the conditionings and we find $\text{cond}_2(A) = 1.0382 \times 10^6$ and $\text{cond}_2(\tilde{A}) = 1.4269$. ⊕

5.5.3 Subdivision Schemes

Solution of Exercise 5.11

1. Assume that $f_0(\alpha) = p(\alpha) = p(2^0\alpha)$ where $p \in \mathbb{P}_1$. Suppose that $f_{n-1}(\alpha) = p(u)$, where $u := \alpha t$ and $t := 2^{-n+1}$. Since p is linear we have by Taylor expansion $p(u+s) = p(u) + sp'(u)$ for any $s \in \mathbb{R}$. Then by (5.8)

$$\begin{aligned} f_n(2\alpha) &= f_{n-1}(\alpha) = p(\alpha 2^{-n+1}) = p(2\alpha 2^{-n}) \\ f_n(2\alpha + 1) &= (1/2 + w)[p(u) + p(u+t)] - w(p(u-t) + p(u+2t)) \\ &= (1/2 + w)[2p(u) + tp'(u)] - w(2p(u) + tp'(u)) \\ &= p(u) + t/2p'(u) = p(u + t/2) = p((2\alpha + 1)2^{-n}), \end{aligned}$$

The result follows by induction.

Now if $p \in \mathbb{P}_3$ and $w = 1/16$, we extend the previous computation:

$$\begin{aligned} 16f_n(2\alpha + 1) &= 9[p(u) + p(u+t)] - [p(u-t) + p(u+2t)] \\ &= 9[2p(u) + tp'(u) + \frac{t^2}{2}p''(u) + \frac{t^3}{6}p^{(3)}(u)] \\ &\quad - [2p(u) + tp'(u) + \frac{5}{2}t^2p''(u) + \frac{7}{6}p^{(3)}(u)] \\ &= 16p(u) + 8tp'(u) + 2t^2p''(u) + \frac{t^3}{3}p^{(3)}(t) \\ &= 16p(u + t/2) = 16p((2\alpha + 1)2^{-n}). \end{aligned}$$

2. Using the definition (5.8), we obtain

$$\begin{aligned}
 & f_n(2\alpha + 1) - f_n(2\alpha) \\
 = & \left(\frac{1}{2} + w \right) (f_{n-1}(\alpha) + f_{n-1}(\alpha + 1)) - w(f_{n-1}(\alpha - 1) + f_{n-1}(\alpha + 2)) \\
 & - f_{n-1}(\alpha) \\
 = & \frac{1}{2} [f_{n-1}(\alpha + 1) - f_{n-1}(\alpha)] \\
 + & w[f_{n-1}(\alpha) - f_{n-1}(\alpha - 1)] + w[f_{n-1}(\alpha + 1) - f_{n-1}(\alpha + 2)],
 \end{aligned}$$

which is (5.9). Using $f_n(2\alpha + 2) = f_{n-1}(\alpha + 1)$, similarly, we obtain (5.10).

3. From (5.9) and (5.10), we deduce that

$$\begin{aligned}
 |f_n(2\alpha) - f_n(2\alpha + 1)| & \leq \left(\frac{1}{2} + 2|w| \right) M_{n-1} \\
 |f_n(2\alpha + 1) - f_n(2\alpha + 2)| & \leq \left(\frac{1}{2} + 2|w| \right) M_{n-1}
 \end{aligned}$$

so that $M_n \leq \theta M_{n-1}$ with $\theta = \frac{1}{2} + 2|w|$ from which we deduce $M_n \leq \theta^n M_0$.

4. The function φ_n is a linear function on $[\alpha 2^{-n}, (\alpha + 1)2^{-n}]$, while φ_{n+1} is linear on $[\alpha 2^{-n}, (\alpha + 1/2)2^{-n}]$ and $[(\alpha + 1/2)2^{-n}, (\alpha + 1)2^{-n}]$. At the two endpoints, $t := \alpha 2^{-n}$ or $t := (\alpha + 1)2^{-n}$, we have $\varphi_{n+1}(t) = \varphi_n(t) = f_n(t2^n)$. We deduce that the maximum of $|\varphi_{n+1}(t) - \varphi_n(t)|$ is obtained at the midpoint of the interval, namely $\bar{t} = (\alpha + 1/2)2^{-n} = (2\alpha + 1)2^{-(n+1)}$ where

$$\begin{aligned}
 \varphi_{n+1}(\bar{t}) & = f_{n+1}(2\alpha + 1) \\
 & = \left(\frac{1}{2} + w \right) (f_n(\alpha) + f_n(\alpha + 1)) - w(f_n(\alpha - 1) + f_n(\alpha + 2)) \\
 \varphi_n(\bar{t}) & = \frac{f_n(\alpha) + f_n(\alpha + 1)}{2}.
 \end{aligned}$$

Thus

$$|\varphi_{n+1}(\bar{t}) - \varphi_n(\bar{t})| \leq w|f_n(\alpha) - f_n(\alpha - 1)| + w|f_n(\alpha + 1) - f_n(\alpha - 2)|$$

then we obtain (5.11) for any $t \in \left[\frac{\alpha}{2^n}, \frac{\alpha + 1}{2^n} \right]$.

5. From (5.11), we deduce that $\|\varphi_{n+1} - \varphi_n\|_\infty \leq 2|w|M_n \leq 2|w|\theta^n M_0$. Thus the series $\sum_{k=0}^{+\infty} (\varphi_{k+1} - \varphi_k)$ is absolute convergent and has a continuous sum as soon as $\theta < 1$. It follows that the sequence $\varphi_n = \varphi_0 + \sum_{k=0}^{n-1} (\varphi_{k+1} - \varphi_k)$ convergence to a function $\varphi \in C(\mathbb{R})$. For a given $\alpha \in \mathbb{R}$ and $n \in \mathbb{N}$, we notice that $f_{n+p}(\alpha 2^p) = f_n(\alpha)$ so that $\varphi_{n+p}(\alpha 2^{-n}) = \varphi_{n+p}(\alpha 2^p 2^{-(n+p)}) = f_{n+p}(\alpha 2^p) = f_n(\alpha)$. When p tends to $+\infty$, we deduce that $\varphi(\alpha 2^{-n}) = f_n(\alpha)$.

6. Programs:

- a. f_0 has a period $T \in \mathbb{N}$. Suppose that f_n has a period 2^nT , then for any $\alpha \in \mathbb{Z}$ we obtain

$$\begin{aligned} f_{n+1}(2\alpha + 2^{n+1}T) &= f_{n+1}(2(\alpha + 2^nT)) \\ &= f_n(\alpha + 2^nT) = f_n(\alpha) = f_{n+1}(2\alpha), \end{aligned}$$

and

$$\begin{aligned} f_{n+1}(2\alpha + 1 + 2^{n+1}T) &= f_{n+1}(2(\alpha + 2^nT) + 1) \\ &= \left(\frac{1}{2} + w\right)[f_n(\alpha + 2^nT) + f_n(\alpha + 2^nT + 1)] \\ &\quad - w[f_n(\alpha + 2^nT - 1) + f_n(\alpha + 2^nT + 2)] \\ &= \left(\frac{1}{2} + w\right)[f_n(\alpha) + f_n(\alpha + 1)] \\ &\quad - w[f_n(\alpha - 1) + f_n(\alpha + 2)] = f_{n+1}(2\alpha + 1). \end{aligned}$$

The periodicity now follows by induction.

- b. `pt4iterat.m`

```
function f1=pt4iterat(f0 ,w)
T=size(f0 ,2);
f1 (: ,2:2:2*T)=f0 (: ,1:T);
f1 (: ,1)=(1/2+w)*(f0 (: ,T)+f0 (: ,1))-w*(f0 (: ,T-1)+f0 (: ,2));
f1 (: ,3)=(1/2+w)*(f0 (: ,1)+f0 (: ,2))-w*(f0 (: ,T)+f0 (: ,3));
f1 (: ,2*T-1)=(1/2+w)*(f0 (: ,T-1)+f0 (: ,T))-w*(f0 (: ,T-2)...
+ f0 (: ,1));
f1 (: ,5:2:2*T-3)=(1/2+w)*( f0 (: ,2:T-2)+f0 (: ,3:T-1))...
-w*(f0 (: ,1:T-3)+f0 (: ,4:T));
```

`pt4niterat.m`

```
function f=pt4niterat(f0 ,w,n)
g=f0 ;
for i=1:n
    f=pt4iterat(g,w);
    g=f;
end
```

- c. `pt4getandplot.m`

```
w=1/16;n=0;
% Getting the Data
axis([0 10 0 10])
hold on
axis off
f0 = [];
j = 0;
% Loop, picking up the points.
LW='Left mouse button picks points ,';
RW= ' Right mouse button picks last point .';
```

```

title ([LW,RW])
but = 1;
while but == 1
    [xj ,yj ,but] = ginput(1);
    plot(xj ,yj , 'rx')
    j = j+1;
    f0(:,j) = [xj;yj];
end
f0p=[f0 ,f0 (:,1)];

% Computing and plotting
vprof=uicontrol('Style','slider','Min',-1, 'Max',10, ...
    'Position',[10 5 20 150], 'value',n);
uicontrol('Style','text','Position',[5 155 30 15], ...
    'String','step','BackgroundColor',get(gcf, 'color'));
valw=uicontrol('Style','slider','Min',-0.5, 'Max',1, ...
    'Position',[50 5 20 150], 'value',w);
uicontrol('Style','text','Position',[45 155 30 15], ...
    'String','w','BackgroundColor',get(gcf, 'color'));

while n>-1
    g=f0;
    for i=1:n
        f=pt4iterat(g,w);
        g=f;
    end
    g=[g,g(:,1)];
    plot(f0p(1,:),f0p(2,:),'—b',g(1,:),g(2,:),'r')
    title(['step: ',int2str(n),', w = ',num2str(w)], ...
        'Fontsize',16)
    axis([0 10 0 10]),hold off,axis off
    pause(1)
    n=fix(get(vprof , 'value')); w=get(valw , 'value');
    clear f, clear g
end

```

5.5.4 A Nonlinear Pendulum

Solution of Exercise 5.12

1. For all $i = 2, \dots, n + 1$, the equation in (P) gives $-u''(t_i) + \sin(u(t_i)) = f(t_i)$. With the approximation of $u''(t_i)$, we deduce that

$$-\frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + \sin(v_i) = f(t_i),$$

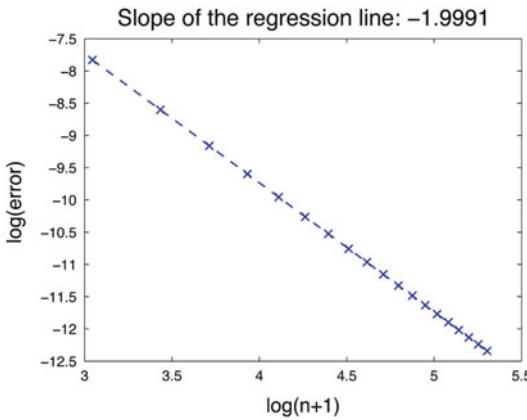


Fig. 5.8 The error is of order 2

and when multiply by h^2 :

$$\begin{aligned}
 -v_1 + 2v_2 - v_3 + h^2 \sin(v_2) &= h^2 f(x_2) \\
 &\vdots \quad \vdots \\
 -v_{i-1} + 2v_i - v_{i+1} + h^2 \sin(v_i) &= h^2 f(x_i) \\
 &\vdots \quad \vdots \\
 -v_n + 2v_{n+1} - v_{n+2} + h^2 \sin(v_{n+1}) &= h^2 f(x_{n+1})
 \end{aligned}$$

Since $v_1 = \alpha$ and $v_{n+2} = \beta$, the approximate problem can be written $\mathbf{Av} = h^2(\mathbf{f} - \sin \mathbf{v}) - \mathbf{b}$.

2. pendul1.m

```
a=0;b=1;
n=3;
h=(b-a)/(n+1);
t=(a:h:b)',
```

3. f1.m

```
function y=f1(t)
y=sin(sin(pi*(t+1/2))-pi^2*sin(pi*(t+1/2));
```

u1.m

```
function y=u1(t)
y=sin(pi*(t+1/2));
```

4. pendul2.m

```
a=0;b=1;
alpha=1;beta=-1;
n=3;
h=(b-a)/(n+1);
t=(a:h:b)';
f=f1(t(2:n+1))
vb=zeros(n,1);
vb(1)=alpha;vb(n)=beta
bu=u1(t)
```

5. pendul3.m

```
pendul2
A=-2*eye(n)+diag(ones(1,n-1),1)+diag(ones(1,n-1),-1)
```

6. pendul4.m

```
pendul3;
p=6;
v=zeros(n,1);
for iter=1:p
    w=A\h^2*(f-sin(v))-vb;
    erriter(iter)=norm(w-v, inf);
    v=w;
end
bv=[alpha;v;beta]
erriter
```

7. pendul5.m

```
a=0;b=1;
alpha=1;beta=-1;
n=5;
h=(b-a)/(n+1);
t=(a:h:b)';
f=f1(t(2:n+1));
vb=zeros(n,1);
vb(1)=alpha;vb(n)=beta;
bu=u1(t);
A=-2*eye(n)+diag(ones(1,n-1),1)+diag(ones(1,n-1),-1);
p=6;
v=zeros(n,1);
for iter=1:p
    w=A\h^2*(f-sin(v))-vb;
    erriter(iter)=norm(w-v, inf);
    v=w;
end
bv=[alpha;v;beta]
error=norm(bv-bu, inf)
tt=a:(b-a)/500:b;
plot(t,bv,'o--',t, bu, 'xr', tt, u1(tt), 'r')
xlabel('t')
ylabel('u')
```

8. pendulumorder.m

```

a=0;b=1;
alpha=1;beta=-1;
arrn=20:10:200;
for k=1:length(arrn)
    n=arrn(k);
    h=(b-a)/(n+1);
    t=(a:h:b)';
    f=f1(t(2:n+1));
    vb=zeros(n,1);
    vb(1)=alpha;vb(n)=beta;
    bu=u2(t);
    A=-2*eye(n)+diag(ones(1,n-1),1)+diag(ones(1,n-1),-1);
    v=zeros(n,1);
    for iter=1:2*n
        w=A\((h^2*(f-sin(v))-vb));
        erriter(iter)=norm(w-v,inf);
        v=w;
    end
    bv=[alpha;v;beta];
    E(k)=norm(bv-bu,inf);
end
plot(log(arrn+1),log(E),'x—')
c=polyfit(log(arrn+1),log(E),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)');
ylabel('log(error)');

```

The computed order is 2.

9. f2.m

```

function y=f2(t)
y=sin(cos(pi*(t+1/2)))-pi*sin(pi*(t+1/2))-pi^2*cos(pi*(t+1/2));

```

u2.m

```

function y=u1(t)
y=cos(pi*(t+1/2));

```

damppendul.m

```

a=0;b=1;
alpha=0;beta=0;
n=5
h=(b-a)/(n+1);
t=(a:h:b)';
f=f2(t(2:n+1));
vb=zeros(n,1);
vb(1)=alpha;vb(n)=beta;
bu=u2(t);
A=-2*eye(n)+diag(ones(1,n-1),1)+diag(ones(1,n-1),-1);
C=+diag(ones(1,n-1),1)-diag(ones(1,n-1),-1);
d=zeros(n,1);

```

```

d(1)=- alpha ;d(n)=beta ;
p=6;
v=zeros(n,1);
for iter=1:p
    w=(A+h/2*C)\(h^2*(f-sin(v))-h/2*d-vb);
    erriter(iter)=norm(w-v, inf);
    v=w;
end
bv=[ alpha ;v;beta ]
error=norm(bv-bu, inf)
tt=a:(b-a)/500:b;
plot(t,bv,'o--',t,bu,'xr',tt,u2(tt),'r')
xlabel('t')
ylabel('u')

```

10. damppendulorder.m

```

a=0;b=1;
alpha=1;beta=-1;
arrn=20:10:200;
for k=1:length(arrn)
    n=arrn(k);
    h=(b-a)/(n+1);
    t=(a:h:b)';
    f=f1(t(2:n+1));
    vb=zeros(n,1);
    vb(1)=alpha;vb(n)=beta;
    bu=u1(t);
    A=-2*eye(n)+diag(ones(1,n-1),1)+diag(ones(1,n-1),-1);
    v=zeros(n,1);
    for iter=1:2*n
        w=A\((h^2*(f-sin(v))-vb));
        erriter(iter)=norm(w-v, inf);
        v=w;
    end
    bv=[ alpha ;v;beta];
    E(k)=norm(bv-bu, inf);
end
plot(log(arrn+1),log(E),'x--')
c=polyfit(log(arrn+1),log(E),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)');
ylabel('log(error)');

```

Again, the computed order is 2 (Fig. 5.8).

Chapter 6

Polynomial Interpolation

Polynomial interpolation is an important tool in numerical analysis. One of its main uses is to derive formulas used in scientific computing, in particular, almost all formulas for numerical differentiation and integration are derived using interpolation polynomials, for examples, see Chaps. 9 and 12.

Suppose we have a function f that is known at certain points. In **Lagrange interpolation** we want to find a function g which agrees with f at the points and hopefully approximate f at other points. Also, one often wants an upper bound for the error that is obtained with a bound on some derivative of f . We also consider **Hermite interpolation**, where the function g should agree with f and consecutive derivative of f up to a certain order at all the sites. The interpolant g can be a trigonometric polynomial, a sum of exponentials or some other class of suitable functions. Algebraic polynomials are quite convenient and we will use such functions in this chapter.

To compute with polynomials, we need a basis for the space of polynomials of a given degree. Different bases lead to different representations of the interpolation polynomial and the choice can affect the accuracy of the computation in floating point arithmetic. In this chapter, we consider several useful bases and derive ways to convert from one basis to another. In Exercise 6.1, we consider the quadratic case using three different bases, while the general case is treated in Exercises 6.3 and Sect. 6.2.2. The error term is derived in Exercises 6.6 and 6.7. In Exercise 6.2, the cubic case for Hermite interpolation is studied, using again three different bases while in Exercises 6.4 and 6.5, we are back to the general Lagrange case.

When the number of sampling points of f increases, even in a given interval, it can happen that the error between f and g does not go to 0 (Exercise 6.8). Using piecewise polynomials can be a good alternative, see Chap. 8.

In Exercise 6.9, we propose a new stable basis for the Lagrange polynomial which is used in Exercise 6.10 for an interpolation of a sampled function of 2 variables. These exercises give an opportunity to use the MATLAB function `meshgrid` which is described in Sect. 1.5.

As mentioned before, applications are proposed in other chapters, but also in this chapter, there is an additional exercise treating the approximation of derivatives followed by a differential equation (Exercises 6.11).

Review:

- **Polynomials.** For an integer $n \geq 0$, we denote by $\mathbb{P}_n = \text{span}(1, t, \dots, t^n)$ the space of polynomials of degree at most n . \mathbb{P}_n is an $n + 1$ dimensional vector space and a set $\{p_1, \dots, p_{n+1}\}$ of $n + 1$ polynomials in \mathbb{P}_n is a **basis** for \mathbb{P}_n if and only if the elements are linearly independent, i.e., if $\sum_{j=1}^{n+1} c_j p_j(t) = 0$ for all t in a set containing at least $n + 1$ distinct points, then $c_1 = \dots = c_{n+1} = 0$. If $\{p_1, \dots, p_{n+1}\}$ is a basis for \mathbb{P}_n then every $p \in \mathbb{P}_n$ can be written in the form $p(t) = \sum_{j=1}^{n+1} c_j p_j(t)$ and the coefficients c_1, \dots, c_{n+1} are uniquely given. The set of powers $\{1, t, \dots, t^n\}$ is the **power basis** for \mathbb{P}_n .

©Comment: Note that in the MATLAB command `polyfit` the coefficients are indexed in opposite order $p(t) = c_1 t^n + c_2 t^{n-1} + \dots + c_n t + c_{n+1}$. ☺

- **Lagrange interpolation.**

Joseph-Louis Lagrange (1736–1813), born Giuseppe Lodovico (Luigi) Lagrangia, was a mathematician and astronomer born in Turin, Piemont, who lived part of his life in Prussia and part in France. He made significant contributions to all fields of analysis, number theory, and classical and celestial mechanics.



Suppose we have a function $f : [a, b] \rightarrow \mathbb{R}$ that is only known at certain **sites** $\mathbf{x} = (x_1, \dots, x_{n+1})^T$ with $a = x_1 < x_2 \dots < x_{n+1} = b$. There is a unique polynomial $p \in \mathbb{P}_n$ such that $p(x_i) = y_i := f(x_i)$, $i = 1, \dots, n + 1$. If $f \in C^{n+1}([a, b])$ there exists to each $t \in [a, b]$ a number $\xi \in (a, b)$ such that

$$f(t) = p(t) + \frac{1}{(n+1)!} \prod_{j=1}^{n+1} (t - x_j) f^{(n+1)}(\xi). \quad (6.1)$$

- **Hermite interpolation.**

Charles Hermite (1822–1901) was a French mathematician who did research on number theory, quadratic forms, invariant theory, orthogonal polynomials, elliptic functions, and algebra. Hermite polynomials, Hermite interpolation, Hermite normal form, Hermitian operators, and cubic Hermite splines are named in his honor.



Suppose in addition to values we also know derivatives of f at $n + 1$ sites. More precisely, for $n + 1$ distinct given real numbers $a = x_1 < x_2 \dots < x_{n+1} = b$ and $n + 1$ positive integers $\alpha_1, \dots, \alpha_{n+1}$. Let $k := n + \alpha_1 + \dots + \alpha_{n+1}$. If f is a function defined on $[a, b]$ with α_i derivatives at the point x_i , $i = 1, \dots, n + 1$, then there exists a unique polynomial $p \in \mathbb{P}_k$ such that

$$p^{(j)}(x_i) = f^{(j)}(x_i) \text{ for } j = 0, 1, \dots, \alpha_i \text{ and } i = 1, \dots, n + 1. \quad (6.2)$$

- **Chebyshev and uniform sites.**

In the exercises, the sites for interpolation of degree n will often be one of the following two kinds relative to an interval $[a, b]$. The **Chebyshev sites** are given by

$$x_i = \frac{a+b}{2} + \frac{a-b}{2} \cos\left(\pi\left(\frac{i-1}{n}\right)\right), \quad i = 1, 2, \dots, n+1, \quad (6.3)$$

while the **uniform sites** are defined as

$$u_i = a + (i-1) \frac{b-a}{n}, \quad i = 1, 2, \dots, n+1. \quad (6.4)$$

6.1 Some Special Cases

Exercise 6.1. Quadratic Lagrange interpolation.

Given sites $a < b < c$ and y_a, y_b, y_c in \mathbb{R} , we want to find a polynomial $p \in \mathbb{P}_2$ such that $p(a) = y_a, p(b) = y_b, p(c) = y_c$.

1. Let $\ell_a(t) := \frac{(t-b)(t-c)}{(a-b)(a-c)}, \ell_b(t) := \frac{(t-c)(t-a)}{(b-c)(b-a)}, \ell_c(t) := \frac{(t-a)(t-b)}{(c-a)(c-b)}$. Show that $\{\ell_a, \ell_b, \ell_c\}$ is a basis for \mathbb{P}_2 .
2. The previous basis is known as the **quadratic Lagrange basis**, and p , written in this form, is called the **quadratic Lagrange form** of the interpolation polynomial. Show existence and uniqueness of p by using this basis .
3. Compute p using the **quadratic power basis** $\{1, t, t^2\}$.
4. Compute p using the **quadratic Newton basis** $\{1, (t-a), (t-a)(t-b)\}$. This form of p is called the **quadratic Newton form** of p .

Sir Isaac Newton (1642–1727) was an English physicist, mathematician, astronomer, philosopher, chemist and theologian. In mathematics, he worked for the development of differential and integral calculus and also demonstrated the generalised binomial theorem, developed Newton's method for approximating the roots of a function, and contributed to the study of power series.



Exercise 6.2. Cubic Hermite interpolation.

We start with values and first derivatives y_0, y'_0, y'_1, y_1 at the two sites 0, 1, and the aim of the problem is to find an interpolation polynomial $p \in \mathbb{P}_3$ such that $p(k) = y_k, p'(k) = y'_k$ for $k = 0, 1$. The polynomial p can be computed using any basis for \mathbb{P}_3 and we consider 3 possible choices.

1. Compute the analytic expression for the following 4 functions $H_i^3(t) \in \mathbb{P}_3$ satisfying the interpolation conditions

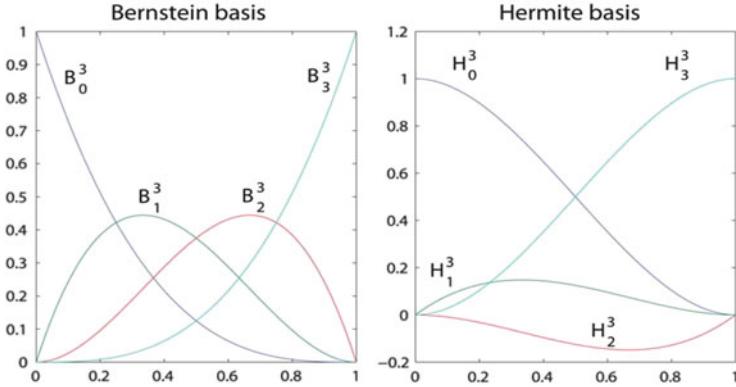


Fig. 6.1 The Bernstein and Hermite basis for \mathbb{P}_3

$$\begin{aligned} H_0^3(0) &= 1, (H_0^3)'(0) = 0, (H_0^3)'(1) = 0, H_0^3(1) = 0, \\ H_1^3(0) &= 0, (H_1^3)'(0) = 1, (H_1^3)'(1) = 0, H_1^3(1) = 0, \\ H_2^3(0) &= 0, (H_2^3)'(0) = 0, (H_2^3)'(1) = 1, H_2^3(1) = 0, \\ H_3^3(0) &= 0, (H_3^3)'(0) = 0, (H_3^3)'(1) = 0, H_3^3(1) = 1. \end{aligned}$$

2. Show that $\{H_i^3\}_{i=0,1,2,3}$ is a basis for \mathbb{P}_3 . It is called the **cubic Hermite basis**.
3. Write down the interpolation polynomial p using this basis.
4. We define $B_0^3(t) = (1-t)^3$, $B_1^3(t) = 3t(1-t)^2$, $B_2^3(t) = 3t^2(1-t)$, $B_3^3(t) = t^3$ for $t \in \mathbb{R}$. Show that $\{B_i^3\}_{i=0,1,2,3}$ is a basis for \mathbb{P}_3 . It is called the **cubic Bernstein basis**.
5. Write expressions for $[B_i^3]_{i=0,1,2,3}$ in terms of the power basis. Thus, find a matrix $A \in \mathbb{R}^{4 \times 4}$ such that

$$[B_0^3(t) \quad B_1^3(t) \quad B_2^3(t) \quad B_3^3(t)] = [1 \quad t \quad t^2 \quad t^3] A.$$

6. Similarly, write $[H_i^3]_{i=0,1,2,3}$ in terms of the power basis using a matrix B .
7. Use the matrices A, B to find the expression for $[H_i^3]_{i=0,1,2,3}$ in terms of the Bernstein basis.
8. Find the interpolant p using the Bernstein basis.
9. Use MATLAB to plot the polynomials $\{B_i^3\}_{i=0,1,2,3}$ and $\{H_i^3\}_{i=0,1,2,3}$ on $[0, 1]$ (Fig. 6.1).

6.2 Lagrange Interpolation

We now generalize Exercise 6.1 to arbitrary degree n . We are given sites $x = (x_1, \dots, x_{n+1}) \in \mathbb{R}^{n+1}$ with $a := x_1 < x_2 < \dots < x_{n+1} =: b$ and data $y_i = f(x_i)$ for $i = 1, \dots, n+1$.

6.2.1 The Lagrange Basis

In the following exercise, we consider the Lagrange basis and use it to find the interpolation polynomial. Also uniqueness is shown.

Exercise 6.3. The Lagrange form

Define

$$p := \sum_{k=1}^{n+1} y_k \ell_k \text{ where } \ell_k(t) := \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{t - x_j}{x_k - x_j}, \quad k = 1, \dots, n+1. \quad (6.5)$$

1. Show that $\ell_k(x_i) = \delta_{i,k} = \begin{cases} 1, & \text{if } i = k, \\ 0, & \text{otherwise,} \end{cases}$, $p \in \mathbb{P}_n$, and that $p(x_i) = y_i$ for $i = 1, \dots, n+1$.
2. Show that $\{\ell_1, \ell_2, \dots, \ell_{n+1}\}$ is a basis for \mathbb{P}_n . This is the **Lagrange basis** for \mathbb{P}_n .
3. Show that the polynomial p defined by (6.5) is the only solution of the interpolation problem. We call it the **Lagrange form** of the interpolation polynomial.

6.2.2 The Newton Form

In this section, we generalize the Newton form of the Lagrange interpolation polynomial to any degree n .

Consider the sequence of interpolation polynomials $p_0, p_1, p_2, \dots, p_n$, where p_k is the Lagrange polynomial of degree at most k such that $p_k(x_i) = y_i = f(x_i)$ for $i = 1, \dots, k+1$ and $k = 0, 1, \dots, n$.

When using the Lagrange basis, a completely new computation has to be performed when adding a new point and going from p_{k-1} to p_k .¹ The Newton basis and divided differences will give an alternative computation of $p = p_n$.

Exercise 6.4. Newton basis

1. Show that $\left\{1, (t - x_1), \dots, \prod_{j=1}^n (t - x_j)\right\}$ is a basis of \mathbb{P}_n . This is known as the **Newton basis**.
2. Show that for $k = 1, \dots, n$,

$$p_k(t) = p_{k-1}(t) + [x_1, \dots, x_{k+1}]f \times (t - x_1) \dots (t - x_k),$$

where $[x_1, \dots, x_{k+1}]f$ is the coefficient of t^k in $p_k(t)$.

Comment: The interest of the notation $[x_1, \dots, x_{k+1}]f$ will become clear in the following exercise. ☺

¹ However, see Exercise 6.9.

3. Deduce the Newton form

$$p(t) = [x_1]f + [x_1, x_2]f \times (t - x_1) + \cdots + [x_1, \dots, x_{n+1}]f \times (t - x_1) \cdots (t - x_n) \quad (6.6)$$

where $[x_1]f = y_1$.

Exercise 6.5. Divided differences

Define for $j = 1, \dots, n+1-k$, $k = 0, 1, \dots, n$ the interpolation polynomials $p_{j,k} \in \mathbb{P}_k$ by $p_{j,k}(x_i) = f(x_i)$ for $i = j, j+1, \dots, j+k$. Thus $p_{j,0}(t) = f(x_j)$ and $p_{1,n} = p_n = p$. Also denote the t^k coefficients of $p_{j,k}$ by $[x_j, \dots, x_{j+k}]f$.

1. Show that for $j = 1, \dots, n+1-k$, $k = 1, \dots, n$,

$$p_{j,k}(t) = \frac{x_{j+k} - t}{x_{j+k} - x_j} p_{j,k-1}(t) + \frac{t - x_j}{x_{j+k} - x_j} p_{j+1,k-1}(t). \quad (6.7)$$

▷ *Math Hint*² ◁

2. Deduce that for $j = 1, \dots, n+1-k$, $k = 1, \dots, n$,

$$[x_j, \dots, x_{j+k}]f = \frac{[x_{j+1}, \dots, x_{j+k}]f - [x_j, \dots, x_{j+k-1}]f}{x_{j+k} - x_j}, \quad (6.8)$$

where $[x_{j+1}, \dots, x_{j+k}]f$ and $[x_j, \dots, x_{j+k-1}]f$ are the t^{k-1} coefficients of $p_{j+1,k-1}$ and $p_{j,k-1}$, respectively. ▷ *Math Hint*³ ◁

The number $[x_j, \dots, x_{j+k}]f$ is called a **divided difference of f of order k** .

3. **Computation of $[x_j, \dots, x_{j+k}]f$** : For simplicity drop f and write $[x_j, \dots, x_{j+k}]$ instead of $[x_j, \dots, x_{j+k}]f$. A divided difference scheme for $n = 4$ is shown in Table 6.1. The divided differences needed in the Newton form of the interpolation polynomial (6.6) are found along the upper NW to SE diagonal.

- (a) Construct the difference scheme corresponding to $x = (-2, -1, 0, 1, 2)$ and $y = (-1, 1, -1, 1, -1)$ and write down the Newton form of the interpolation polynomial p of degree 4 interpolating these values. ▷ *Math Hint*⁴ ◁
- (b) What is the power form of p ?

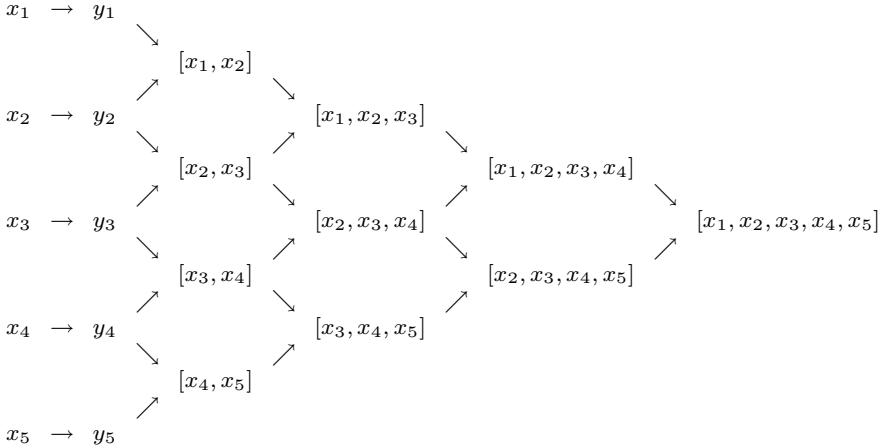
6.2.3 The Error Term

Go directly to Exercise 6.7 if you prefer to skip the proof of the error term (6.1) in Exercise 6.6.

² Use induction to show that $p_{j,k} \in \mathbb{P}_k$ and that $p_{j,k}(x_i) = f(x_i)$ for $i = j, \dots, j+k$.

³ The t^k coefficients on both sides of (6.7) must be the same.

⁴ Use the form in Table 6.1.

Table 6.1 A divided difference scheme

Exercise 6.6. Define $e := f - p$. For fixed $t \in [a, b] \setminus \{x_1, \dots, x_{n+1}\}$, we define the function $g : [a, b] \rightarrow \mathbb{R}$ by

$$g(x) = e(x) - e(t) \prod_{j=1}^{n+1} \frac{(x - x_j)}{(t - x_j)}.$$

1. Show that $g(x_1) = g(x_2) = \dots = g(x_{n+1}) = 0$ and $g(t) = 0$. Deduce that there exist $a < x_1^1 < x_2^1 < \dots < x_{n+1}^1 < b$ so that

$$g'(x_1^1) = g'(x_2^1) = \dots = g'(x_{n+1}^1) = 0.$$

▷ *Math Hint*⁵ ◁

2. Show that there exists a point $x_1^{n+1} = \xi$ in (a, b) so that $g^{(n+1)}(\xi) = 0$.
3. Show (6.1). ▷ *Math Hint*⁶ ◁

Exercise 6.7. Error in linear interpolation

Consider the case $n = 1$ ($x_1 = a$, $x_2 = b$) in (6.1). We can deduce that

$$|f(t) - p(t)| \leq \frac{(b-a)^2}{8} \max_{a \leq z \leq b} |f''(z)|, \quad t \in [a, b] \quad (6.9)$$

by bounding $|(t-a)(t-b)|$ by $(b-a)^2/4$. In the following, we give a different proof.

⁵ Use Rolle's theorem.

⁶ Use that $p^{(n+1)}(\xi) = 0$.

1. Give the Taylor expansion of $f(a)$ at t of order 1 with remainder.

Brook Taylor (1685–1731) was an English mathematician who obtained a solution of the problem of the “center of oscillation”. To determine the form of movement of a vibrating string, he created the calculus of finite differences and its work contained the celebrated formula known as Taylor’s theorem, the importance of which remained unrecognized until 1772, when J. L. Lagrange realized its powers.



2. Similarly, give the expansion of $f(b)$ at t .
 3. If $M_2 = \max_{a \leq z \leq b} |f''(z)|$, and $t \in [a, b]$ deduce

$$|(b-t)f(a) + (t-a)f(b) - (b-a)f(t)| \leq \frac{(t-a)(b-t)}{2}(b-a)M_2.$$

4. Prove (6.9)

6.2.4 The Runge Phenomenon

We program an example where interpolation at uniform sites gives poor results for the approximation near the end of the range. This example also demonstrates that the error near the ends does not go to 0 when the number of sampling points increases.

Exercise 6.8. Non convergent error

We interpolate the function given by $f(t) = \frac{1}{1+t^2}$ at $n+1$ uniform sites in the interval $[-5, 5]$. More specifically $x_j = -5 + 10 \frac{j-1}{n}$ for $j = 1, \dots, n+1$.

Write a MATLAB program to make a table that contains a sequence of values of n together with the values of f , p , and $f-p$ at the point $x_{n-\frac{1}{2}} = 5 - 5/n$. \triangleleft MATLAB Hint⁷ \triangleright Test using $n = 2, 4, 6, \dots, 20$. and plot the graph of f and p . See Fig. 6.2 for $n = 10$.

```
>> tabl =
    2.0000    0.1379    0.7596    0.6217
    4.0000    0.0664   -0.3568    0.4232
    6.0000    0.0545    0.6079    0.5534
    8.0000    0.0497   -0.8310    0.8807
   10.0000    0.0471    1.5787    1.5317
   12.0000    0.0454   -2.7550    2.8004
   14.0000    0.0443    5.3327    5.2884
   16.0000    0.0435  -10.1739   10.2174
   18.0000    0.0429   20.1237   20.0808
```

⁷ To obtain the interpolant for some n , one can use the instruction `coeff=polyfit(x,y,n)`, which computes the coefficients of the polynomial p_n of degree n interpolating f at $\mathbf{x} = (x_1, \dots, x_{n+1})^T$. The command `polyval(coeff,t)` computes $p_n(t)$ for one or several values of t .

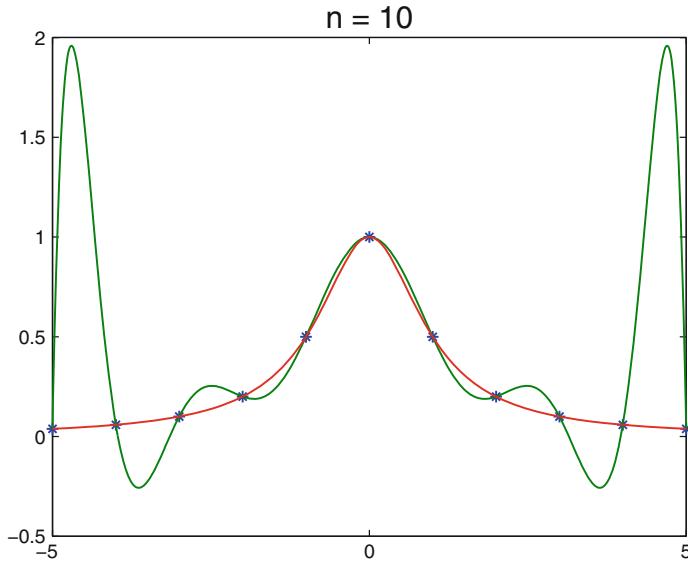


Fig. 6.2 Function and Lagrange interpolant

20.0000	0.0424	-39.9524	39.9949
n	f (t)	p (t)	error

⌚Comment: The polynomial p_n does not approximate f very well. Moreover the approximation gets worse as n increases. We can get better approximation by interpolating at the **Chebyshev sites** given by (6.3), see also Chap. 11. ☺

Pafnuty Lvovich Chebyshev (1821–1894) is considered a founding father of Russian mathematics. He is known for his work in the field of probability, statistics and number theory. (Chebyshev's inequality). Chebyshev polynomials are important in approximation theory. For example the roots of the Chebyshev polynomials of the first kind, which are also called Chebyshev nodes, are used as nodes in polynomial interpolation.



6.3 Stable Lagrange Interpolation

6.3.1 The Univariate Case

In this section we consider an improved formula for the Lagrange form

$$p := \sum_{k=1}^{n+1} y_k \ell_k, \quad \ell_k(t) = \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{t - x_j}{x_k - x_j}, \quad k = 1, \dots, n+1$$

of the interpolation polynomial p . We introduce the quantities

$$\ell(t) := (t-x_1) \cdots (t-x_{n+1}), \quad w_k := \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{1}{x_k - x_j}, \quad k = 1, \dots, n+1, \quad (6.10)$$

Exercise 6.9.

1. Show that the value at t of the polynomial p can be written

$$p(t) = \ell(t) \sum_{k=1}^{n+1} \frac{w_k}{t - x_k} y_k. \quad (6.11)$$

2. We can rewrite (6.11) in a complicated, but more elegant form. Show the **barycentric formula**

$$p(t) = \frac{\sum_{k=1}^{n+1} \frac{w_k}{t - x_k} y_k}{\sum_{k=1}^{n+1} \frac{w_k}{t - x_k}}. \quad (6.12)$$

▷ *Math Hint*⁸ ◁

3. The barycentric form can be updated as efficiently as the Newton form. Suppose the weights w_k^{m-1} for the points x_1, \dots, x_m are computed and we add the point x_{m+1} . Show that the updated weights w_k^m , $k = 1, \dots, m+1$ can be computed in $O(m)$ arithmetic operations.

4. Programs without any loop:

- (a) Write a MATLAB function `matA` with \mathbf{x} as input and the matrix $\mathbf{M} = [a_{ij}]_{i,j=1,r} \in \mathbb{R}^{r \times r}$ for the output where r is the dimension of \mathbf{x} .

```
>> M=matA([1,2,3])
M =
    1     1     1
    2     2     2
    3     3     3
```

- (b) Extend the previous function into `lagweights.m` that computes the weights \mathbf{w} corresponding to the sites \mathbf{x} . ◁ *MATLAB Hint*⁹ ▷

⁸ Use $1 = \sum_{k=1}^{n+1} \ell_k(t) = \ell(t) \sum_{k=1}^{n+1} \frac{w_k}{t - x_k}$.

⁹ Compute the matrix $\mathbf{A} = \mathbf{M} - \mathbf{M}^T + \mathbf{I}$, then compute the matrix \mathbf{B} with elements $1/(x_i - x_j)$ except for the diagonal and finally the $\mathbf{w} = [w_i]_{i=1,r}$ using `sum`.

```
>> W=lagweights([1,2,3]')
```

W =

0.5000
-1.0000
0.5000

In this example,

```
B =
```

1.0000	-1.0000	-0.5000
1.0000	1.0000	-1.0000
0.5000	1.0000	1.0000

- (c) Write a function `specialsum.m` that computes $\sum_{j=1}^r \frac{z_j}{t-x_j}$ where x and z are two vectors of the same dimension r and t can be an array $T=[t_1, \dots, t_s]$. The output is an array of size s . \triangleleft MATLAB Hint¹⁰ \triangleright

Do not worry if one of the t 's is equal to one of x 's. In this case NaN¹¹ is returned.

```
>> S=specialsum((0:3)',(1.5:0.5:3)',-4:0)
```

S =

-1.6202	-2.0000	-2.6417	-4.0833	NaN
---------	---------	---------	---------	-----

In the example the arrays are given by

```
>> X=(0:3)';T=-4:0;[TT,XX]=meshgrid(T,X)
```

TT =

-4	-3	-2	-1	0
-4	-3	-2	-1	0
-4	-3	-2	-1	0
-4	-3	-2	-1	0

XX =

0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

- (d) Write a program `lagpolint.m` that computes the barycentric form of p at the points t . The vectors x and y are given. The weights w are computed using the `lagweights` function. Both sums are obtained using the `specialsum` function. Test `lagpolint.m` by sampling from the function $y = \sqrt{|t|}$ on $[-1, 1]$. Try first 9 uniform points and then 101 Chebyshev points $x_{j+1} = -\cos(j\pi/n)$, $j = 0, 1, \dots, 100 := n$. Plots are shown in Fig. 6.3.

¹⁰ The first step is to define two arrays TT, XX of dimension $r \times s$ from T and X which is done by `[TT,XX]=meshgrid(T,X)`. Details on `meshgrid` are given in Sect. 1.5.

¹¹ NaN is “Not a Number”, for example in the case of a division by 0 followed by a sum.

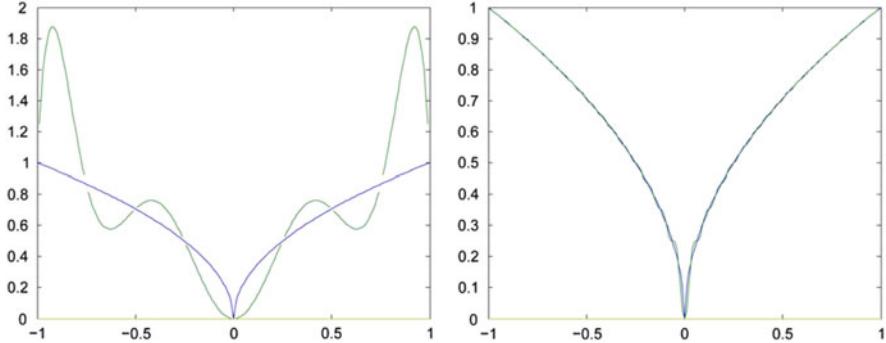


Fig. 6.3 Interpolating $\sqrt{|t|}$ at 9 uniform points in $[-1, 1]$, (left), and at 101 Chebyshev points (right)

6.3.2 Application to Interpolating Surfaces

Exercise 6.10.

We are given a grid $\{(x_i, y_j) : i = 1, \dots, m+1, j = 1, \dots, n+1\}$ with $a := x_1 < x_2 < \dots < x_{m+1} =: b$, $c := y_1 < y_2 < \dots < y_{n+1} =: d$ and data $z_{ij} = f(x_i, y_j)$ for $i = 1, \dots, m+1$ and $j = 1, \dots, n+1$. Let $\mathbf{x} = (x_1, \dots, x_{m+1}) \in \mathbb{R}^{m+1}$ and $\mathbf{y} = (y_1, \dots, y_{n+1}) \in \mathbb{R}^{n+1}$.

For the two variables, with extended notations from the previous exercise, we define

$$\begin{aligned}\ell_k(s) &:= \prod_{\substack{j=1 \\ j \neq k}}^{m+1} \frac{s - x_j}{x_k - x_j}, \quad w_k := \prod_{\substack{j=1 \\ j \neq k}}^{m+1} \frac{1}{x_k - x_j}, \quad k = 1, \dots, m+1 \\ \bar{\ell}_k(t) &:= \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{t - y_j}{y_k - y_j}, \quad \bar{w}_k := \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{1}{y_k - y_j}, \quad k = 1, \dots, n+1.\end{aligned}$$

1. Show that the polynomial P in two variables defined by

$$P(s, t) := \sum_{p=1}^{m+1} \sum_{q=1}^{n+1} z_{pq} \ell_p(s) \bar{\ell}_q(t) \tag{6.13}$$

satisfies $P(x_i, y_j) = z_{ij}$ for $i = 1, \dots, m+1$ and $j = 1, \dots, n+1$.

2. Show that

$$P(s, t) = \frac{\sum_{p=1}^{m+1} \sum_{q=1}^{n+1} \frac{w_p \bar{w}_q}{(s - x_p)(t - y_q)} z_{pq}}{\sum_{p=1}^{m+1} \frac{w_p}{s - x_p} \sum_{q=1}^{n+1} \frac{\bar{w}_q}{t - y_q}}. \tag{6.14}$$

3. Programs:

- (a) To compute the values of a function of 2 variables, write a MATLAB function `f.m` with 2 arrays of same dimension as input and an array as output. Test with $f(s, t) = -20s(1-s) + 0.1t^3$.

```
>>S=[ 0 ,1 ,2]; T=[ -1 ,1];[SS,TT]=meshgrid (T,S)
TT =
    -1      1
    -1      1
    -1      1
SS =
    0      0
    1      1
    2      2
>> f(SS,TT)
ans =
   -0.1000    0.1000
   -0.1000    0.1000
  39.9000   40.1000
```

- (b) Write a program `interpolsurf.m` such that given the grid $\mathbf{x} = (x_1, \dots, x_{n+1})$ and $\mathbf{y} = (y_1, \dots, y_{m+1})$ plus the sampled values $\mathbf{z} = (f(x_i, y_j)) \in \mathbb{R}^{(m+1) \times (n+1)}$, computes the polynomial p , then plot the surfaces $f(s, t)$, $p(s, t)$ and the difference $f - p$. Test with $m = 3$, \mathbf{x} a uniform partition of $[0, \pi]$, $n = 7$, \mathbf{y} a uniform partition of $[0, 2\pi]$, and $f(s, t) = \sin(s + t)$. \triangleleft
- MATLAB Hint¹² \triangleright

6.4 Approximation of Derivatives, Part I

Let $f : [a, b] \rightarrow \mathbb{R}$ be a function sampled at sites $\mathbf{x} = (x_1, \dots, x_{n+1})^T$ with $a \leq x_1 < x_2 < \dots < x_{n+1} \leq b$. We let $\mathbf{y} = f(\mathbf{x}) = (f(x_1), \dots, f(x_{n+1}))^T$ be the corresponding y -values. The goal is to approximate the derivatives of f . For this we will use the values of the derivatives of the Lagrange interpolation polynomial at the sites. More standard finite difference approximations to the derivatives will be considered in Chap. 12.

We will also use these approximations to solve a differential equation in a somewhat nonstandard way. A more systematic treatment of numerical methods for differential equations is given in Chaps. 12 and 13.

¹² Use the functions `lagweights.m` and `specialsum.m` in Exercise 6.9. Since we have 4 input arrays, $\mathbf{x}, \mathbf{y}, \mathbf{u}, \mathbf{v}$ two loops can be useful.

We recall that the Lagrange interpolation polynomial $p \in \mathbb{P}_n$ at $(x_i, y_i)_{i=1}^{n+1}$ is given by

$$p(t) = \sum_{j=1}^{n+1} y_j \ell_j(t) \text{ with } \ell_j(t) = \prod_{\substack{k=1 \\ k \neq j}}^{n+1} \frac{t - x_k}{x_j - x_k}.$$

Exercise 6.11.

1. Show that the vector of the derivatives of p at \mathbf{x} , $\mathbf{d}_f := p'(\mathbf{x}) = (p'(x_1), \dots, p'(x_{n+1}))^T$ satisfies $\mathbf{d}_f = \mathbf{M}\mathbf{y}$ with $\mathbf{M} \in \mathbb{R}^{(n+1) \times (n+1)}$ given by

$$m_{ij} = \ell'_j(x_i) = \begin{cases} \frac{w_j}{w_i(x_i - x_j)} & \text{if } i \neq j \\ \sum_{\substack{k=1 \\ k \neq j}}^{n+1} \frac{1}{(x_j - x_k)} & \text{if } i = j \end{cases}, \quad (6.15)$$

where

$$w_i := \prod_{\substack{k=1 \\ k \neq i}}^{n+1} \frac{1}{x_i - x_k}, \quad i = 1, \dots, n+1,$$

was used in the barycentric formula for the Lagrange interpolant and (6.10).

▷ *Math Hint*¹³ ◁

The vector \mathbf{d}_f contains approximations to the derivatives of f at the points x_i , $i = 1, \dots, n+1$.

2. Let $\mathbf{Q} \in \mathbb{R}^{(n+1) \times (n+1)}$ have elements

$$q_{i,j} := \begin{cases} 1/(x_i - x_j), & \text{if } i \neq j, \\ 0, & \text{otherwise.} \end{cases},$$

let $\mathbf{e} := (1, 1, \dots, 1)^T \in \mathbb{R}^{n+1}$, and $\mathbf{W} := \text{diag}(w_1, \dots, w_{n+1})$. Show that

$$\mathbf{M} = \mathbf{W}^{-1} \mathbf{Q} \mathbf{W} + \text{diag}(\mathbf{Q}\mathbf{e}). \quad (6.16)$$

3. Write a function `mmatrix.m` that computes the matrix \mathbf{M} . ◁ *MATLAB Hint*¹⁴ ▷ with \mathbf{x} as input.

```
>> mmatrix([1,2,3])
ans =
-1.5000    2.0000   -0.5000
-0.5000     0.0000    0.5000
 0.5000   -2.0000    1.5000
```

¹³ For $i \neq j$, differentiate ℓ_j in the form $\ell_j(t) = \ell(t)w_j/(t - x_j)$, where $\ell(t) := (t - x_1) \cdots (t - x_{n+1})$. For $i = j$ set $t = x_j$ in $\frac{d}{dt} \log \ell_j(t) = \ell'_j(t)/\ell_j(t)$ and solve for $\ell'_j(x_j)$.

¹⁴ For \mathbf{W} use the `lagweights` program. \mathbf{Q} is almost a Cauchy matrix. Check `gallery`.

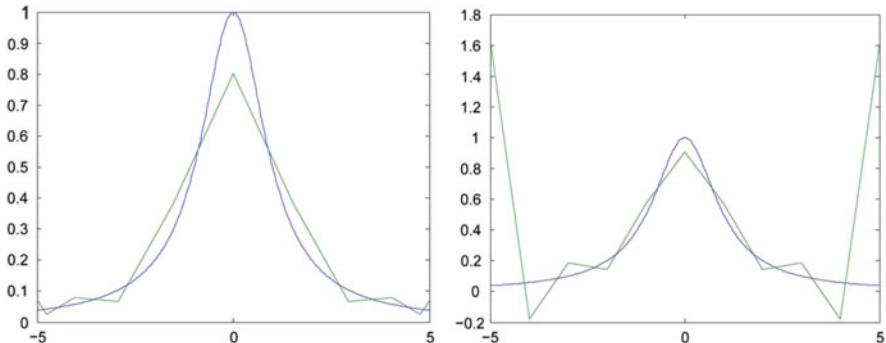


Fig. 6.4 Exact and Approximate Derivatives of $\arctan(t)$ on $[-5, 5]$. 11 Chebyshev points (left) and 11 uniform point (right)

4. Write a function `dy=df(x,y)` that computes d_f . Test with $f(x) = x^2$, $x = (0, 1, 5)$.

```
>> x=[1,2,5];
>> y=x.^2;
>> df(x,y)
ans =
    2
    4
    10
```

Exercise 6.12.

In this exercise we want to see how the choice of interpolation sites can affect the accuracy of the derivatives computed by the method in Exercise 6.11. We consider a function $f : [a, b] \rightarrow \mathbb{R}$ known at the **Chebyshev sites** given by (6.3), or the **uniform sites**, see (6.4).

1. Write a function `plotder(x,f,derf,m)` that makes a plot of the approximate derivatives d_f of f at x together with the exact derivatives defined by the function `derf` using m plot points. Test with $n = 10$, $[a, b] = [-5, 5]$, $f(t) = \arctan(t)$, and $\text{derf}(t) = 1/(1+t^2)$. Try both the Chebyshev- and the uniform sites. The function `derf` should be defined in a separate `m` file (Fig. 6.4)

```
>> x=-5*cos(pi*(0:1/10:1));
>> plotder(x,@atan,@derf,300)
>> u=-5:1:5;
>> plotder(u,@atan,@derf,300)
```

Exercise 6.13. A differential equation

Consider the differential equation $u' + u = 2e^t$. We want to find approximations $\mathbf{y} = (y_1, \dots, y_{n+1})$ to some solution u at sites $\mathbf{x} = (\frac{0}{n}, \frac{1}{n}, \dots, \frac{n}{n})$. Let $\mathbf{M} \in \mathbb{R}^{(n+1) \times (n+1)}$ be the matrix in (6.15). The matrix \mathbf{M} is singular, but $\mathbf{M} + c\mathbf{I}$ is

nonsingular for c sufficiently large. In fact for our case, for n not too large, $c = 1$ suffices.

1. Find a nonzero vector \mathbf{y} such that $M\mathbf{y} = \mathbf{0}$.
2. Explain why the solution \mathbf{y} of the linear system $(M + I)\mathbf{y} = \mathbf{b}$, where $b_i = 2e^{(i-1)/n}$ for $i = 1, 2, \dots, n+1$ can be considered as an approximation to some solution u at x .
3. Write a function `y=ode1(n)` that computes \mathbf{y} in this way. Test with $n = 2$.

```
>> ode1(2)
ans =
    3.6135
    3.2276
    3.6834
```

4. Verify that the solution of $u' + u = 2e^t$, $u(0) = \alpha$ is $u(t) = 2 \sinh(t) + \alpha e^{-t}$. Write a function `e=ode2(n)` that computes the maximum error $\max_j |y_j - u_j|$ corresponding to $\alpha = y_1$. Test for $n = 2, 4, 8$.

```
>> e2=ode2(2);
>> e4=ode2(4);
>> e8=ode2(8);
>> format shortE
>> [e2,e4,e8]
ans =
    6.3007e-03    1.2411e-05    2.1636e-11
```

6.5 Solutions

6.5.1 Some Special Cases

Exercise 6.1

1. The set $\{\ell_a, \ell_b, \ell_c\}$ has three elements all belonging to \mathbb{P}_2 and this space has dimension 3. It is necessary and sufficient to prove the linear independence of these three polynomials. A simple evaluation gives $\ell_a(a) = 1$, $\ell_b(a) = \ell_c(a) = 0$. Suppose that $\lambda_a \ell_a + \lambda_b \ell_b + \lambda_c \ell_c = 0$. When evaluating this polynomial at $t = a$, we obtain $\lambda_a = 0$. Similarly $\lambda_b = \lambda_c = 0$ when evaluating the polynomials at b and c .
2. Clearly $p = y_a \ell_a + y_b \ell_b + y_c \ell_c$ is an answer to the interpolation problem. Unicity is deduced from the fact that any interpolation polynomial expressed in terms of the quadratic Lagrange basis must have the same coefficients y_a, y_b, y_c . Since the coefficients of a basis are unique the interpolation polynomial is unique.

3. Expanding ℓ_a , we find $\ell_a(t) = \frac{(t-b)(t-c)}{(a-b)(a-c)} = \frac{t^2 - (b+c)t + bc}{(a-b)(a-c)}$, and we obtain

$$\begin{bmatrix} \ell_a(t) \\ \ell_b(t) \\ \ell_c(t) \end{bmatrix} = \begin{bmatrix} \frac{bc}{(a-b)(a-c)} & -\frac{b+c}{(a-b)(a-c)} & \frac{1}{(a-b)(a-c)} \\ \frac{ca}{(b-c)(b-a)} & -\frac{c+a}{(b-c)(b-a)} & \frac{1}{(b-c)(b-a)} \\ \frac{ab}{(c-a)(c-b)} & -\frac{a+b}{(c-a)(c-b)} & \frac{1}{(c-a)(c-b)} \end{bmatrix} \times \begin{bmatrix} 1 \\ t \\ t^2 \end{bmatrix}$$

from which we deduce that

$$\begin{aligned} p(t) &= \frac{y_a bc}{(a-b)(a-c)} + \frac{y_b ca}{(b-c)(b-a)} + \frac{y_c ab}{(c-a)(c-b)} \\ &\quad - \left(\frac{y_a(b+c)}{(a-b)(a-c)} + \frac{y_b(c+a)}{(b-c)(b-a)} + \frac{y_c(a+b)}{(c-a)(c-b)} \right) t \\ &\quad + \left(\frac{y_a}{(a-b)(a-c)} + \frac{y_b}{(b-c)(b-a)} + \frac{y_c}{(c-a)(c-b)} \right) t^2. \end{aligned}$$

4. With $p(t) = x_1 1 + x_2(t-a) + x_3(t-a)(t-b)$, the interpolation conditions give

$$\begin{cases} x_1 &= y_a \\ x_1 + x_2(b-a) &= y_b \\ x_1 + x_2(c-a) + x_3(c-a)(c-b) &= y_c \end{cases}$$

which is a triangular system. The solution is

$$x_1 = y_a, \quad x_2 = \frac{y_b - y_a}{b-a}, \quad x_3 = \frac{y_c - 2y_b + y_a}{(b-a)(c-b)(c-a)}.$$

Exercise 6.2

- H_0^3 is a polynomial in \mathbb{P}_3 such that $H_0^3(0) = 1$, $(H_0^3)'(0) = 0$, $(H_0^3)'(1) = 0$, $H_0^3(1) = 0$. Since 1 is a double root we can write $H_0^3(t) = (1-t)^2(at+b)$. The two remaining conditions give two equations for a and b and we obtain $H_0^3(t) = (1-t)^2(2t+1)$. For $H_1^3(t)$, 0 is a root and 1 is a double root and we deduce that $H_1^3(t) = \lambda_1 t(1-t)^2$. From the condition $(H_1^3)'(0) = 1$, $\lambda_1 = 1$, so that $H_1^3(t) = t(1-t)^2$. Similarly, or using symmetry, we obtain $H_2^3(t) = -H_1^3(1-t) = t^2(t-1)$ and $H_3^3(t) = H_0^3(1-t) = t^2(3-2t)$.
- The set $\{H_i^3\}_{i=0,1,2,3}$ has four elements in the four dimensional space \mathbb{P}_3 , and it is sufficient to prove that the polynomials are linearly independent. If $\sum_{i=0}^3 \lambda_i H_i^3 = O$, then at $t = 0$, we obtain $\lambda_0 = 0$. Similarly at $t = 1$, $\lambda_3 = 0$.

When differentiating the remaining sum we find $\sum_{i=1}^2 \lambda_i H_i^{3'} = 0$, then at $t = 0$ and $t = 1$, we find $\lambda_1 = \lambda_2 = 0$.

3. The polynomial interpolating the four values $f(0), f'(0), f'(1), f(1)$ is given by $p = f(0)H_0^3 + f'(0)H_1^3 + f'(1)H_2^3 + f(1)H_3^3$ since this polynomial answers the question and p is unique since $\{H_i^3\}_{i=0,1,2,3}$ is a basis.
4. As before we only need to show linear independence. If $\sum_{i=0}^3 \lambda_i B_i^3(t) = 0$, then evaluating at $t = 0$, we obtain $\lambda_0 = 0$. Similarly at $t = 1$, $\lambda_3 = 0$. Then

$$0 = \sum_{i=1}^2 \lambda_i B_i^3(t) = 3t(1-t)(\lambda_1(1-t) + \lambda_2 t), \quad 0 \leq t \leq 1.$$

It follows that $\lambda_1(1-t) + \lambda_2 t = 0$ for $0 \leq t \leq 1$. Evaluating at $t = 0$ and $t = 1$, we find $\lambda_1 = \lambda_2 = 0$.

5. When expanding the polynomials in the power basis, we obtain

$$\begin{aligned} [B_0^3(t) & B_1^3(t) & B_2^3(t) & B_3^3(t)] = [1 & t & t^2 & t^3] \mathbf{A} \\ \text{where } \mathbf{A} = & \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}. \end{aligned}$$

6. Similarly

$$\begin{aligned} [H_0^3(t) & H_1^3(t) & H_2^3(t) & H_3^3(t)] = [1 & t & t^2 & t^3] \mathbf{B} \\ \text{where } \mathbf{B} = & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & -1 & 3 \\ 2 & 1 & 1 & -2 \end{bmatrix}. \end{aligned}$$

7. Let $\mathcal{H} := [H_0^3(t) \ H_1^3(t) \ H_2^3(t) \ H_3^3(t)]$ and $\mathcal{B} := [B_0^3(t) \ B_1^3(t) \ B_2^3(t) \ B_3^3(t)]$. We deduce that $\mathcal{H} = [1 \ t \ t^2 \ t^3] \mathbf{B} = \mathcal{B} \mathbf{A}^{-1} \mathbf{B}$. Since \mathbf{A} is a lower triangular matrix the inverse \mathbf{A}^{-1} can easily be computed starting from the first row:

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 0 & 0 \\ 1 & 2/3 & 1/3 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{and finally } \mathbf{A}^{-1} \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 0 & 0 \\ 0 & 0 & -1/3 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

8. We find

$$p = \mathcal{H} \begin{bmatrix} f(0) \\ f'(0) \\ f'(1) \\ f(1) \end{bmatrix} = \mathcal{B} \mathbf{A}^{-1} \mathbf{B} \begin{bmatrix} f(0) \\ f'(0) \\ f'(1) \\ f(1) \end{bmatrix} = \mathcal{B} \begin{bmatrix} f(0) \\ f(0) + f'(0)/3 \\ f(1) - f'(1)/3 \\ f(1) \end{bmatrix}.$$

9.

```
t=0:1/500:1;
B0=(1-t).^3; B1=3*t.*((1-t).^2);
B2=3*t.^2.*((1-t)); B3=t.^3;
subplot(1,2,1)
plot(t,B0,t,B1,t,B2,t,B3)
title('Bernstein Basis', 'FontSize', 18)
H0=(t-1).^2.*((1+2*t)); H1=t.*((1-t).^2);
H2=(t-1).*t.^2; H3=t.^2.*((3-2*t));
subplot(1,2,2)
plot(t,H0,t,H1,t,H2,t,H3)
title('Hermite Basis', 'FontSize', 18)
```

6.5.2 Lagrange Interpolation

Exercise 6.3

- For $i = k$ we have $\ell_k(x_k) = \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{x_k - x_j}{x_k - x_j} = 1$. For $i \neq k$ one of the factors in the numerator of $\ell_k(x_i)$ will vanish and $\ell_k(x_i) = 0$. Consequently $p(x_i) = \sum_{k=1}^{n+1} y_k \ell_k(x_i) = \sum_{k=1}^{n+1} y_k \delta_{i,k} = y_i$ for $i = 1, \dots, n+1$. Since each $\ell_k \in \mathbb{P}_n$ it follows that $p \in \mathbb{P}_n$.
- The dimension of \mathbb{P}_n is $n+1$, and $\ell_1, \dots, \ell_{n+1}$ is a set of $n+1$ polynomials in \mathbb{P}_n . We need to show that these polynomials are linearly independent. Suppose $\sum_{k=1}^{n+1} c_k \ell_k(t) = 0$ for $t \in [a, b]$. Inserting $t = x_i$ gives $\sum_{k=1}^{n+1} c_k \ell_k(x_i) = c_i = 0$ for $i = 1, \dots, n+1$ and linear independence follows immediately.
- We already know that p is an answer to the interpolation problem. The basis property of the interpolation polynomial implies uniqueness. For if $p \in \mathbb{P}_n$ then $p = \sum_{k=1}^{n+1} c_k \ell_k$ for some coefficients c_k and if $p(x_i) = y_i$ for $i = 1, \dots, n+1$ then $c_k = y_k$ for all k .

Exercise 6.4

- The dimension of \mathbb{P}_n is $n+1$, and $\left\{1, (t-x_1), \dots, \prod_{j=1}^n (t-x_j)\right\}$ contains $n+1$ polynomials in \mathbb{P}_n . We need to show that these polynomials are linearly independent. Suppose $\sum_{k=1}^{n+1} c_k \prod_{j=1}^{k-1} (t-x_j) = 0$ for $t \in [a, b]$. At $t = x_1$ every element of the sum vanishes except the first one, so that $c_1 = 0$. Then $\sum_{k=2}^{n+1} c_k \prod_{j=1}^{k-1} (t-x_j) = 0$ and at $t = x_2$ we get $c_2 = 0$. By induction $c_i = 0$ for $i = 1, \dots, n+1$.
- For $k = 1, \dots, n$, $p_k(x_j) = p_{k-1}(x_j) = f(x_j)$ for $j = 1, \dots, k$ so that $p_k - p_{k-1}$ is a polynomial of degree at most k that vanishes at the k points $x_j, j = 1, \dots, k$. Thus $p_k(t) - p_{k-1}(t) = a_k \times (t-x_1) \dots (t-x_k)$. $p_k(t)$ and $(t-x_1) \dots (t-x_k)$ are of degree at most k and p_{k-1} is of degree at most $k-1$.

$k - 1$. Hence a_k is the coefficient of t^k in p_k . We use the notation $a_k = [x_1, \dots, x_{k+1}]f$.¹⁵

3. With $[x_1]f = y_1$, we obtain

$$\begin{aligned} p_0(t) &= [x_1]f \\ p_1(t) &= p_0(t) + [x_1, x_2]f \times (t - x_1) \\ &\vdots \\ p_{n-1}(t) &= p_{n-2}(t) + [x_1, \dots, x_n]f \times (t - x_1) \dots (t - x_{n-1}) \\ p_n(t) &= p_{n-1}(t) + [x_1, \dots, x_{n+1}]f \times (t - x_1) \dots (t - x_n). \end{aligned}$$

When summing all the equations, since $p(t) = p_n(t)$, we obtain the result

$$p(t) = [x_1]f + [x_1, x_2]f \times (t - x_1) + \dots + [x_1, \dots, x_{n+1}]f \times (t - x_1) \dots (t - x_n).$$

Exercise 6.5

1. For fixed j, k , denote the righthand side of (6.7) by q .

- We show that $q(x_i) = y_i$ for $i = j, j + 1, \dots, j + k$.

For $j = i$ we have $q(x_i) = p_{j,k-1}(x_i) = y_i$, while for $i = j + 1, \dots, j + k - 1$,

$$\begin{aligned} q(x_i) &= \frac{x_{j+k} - x_i}{x_{j+k} - x_j} p_{j,k-1}(x_i) + \frac{x_i - x_j}{x_{j+k} - x_j} p_{j+1,k-1}(x_i) \\ &= \frac{x_{j+k} - x_i}{x_{j+k} - x_j} y_i + \frac{x_i - x_j}{x_{j+k} - x_j} y_i \\ &= y_i. \end{aligned}$$

Finally $q(x_{j+k}) = p_{j+1,k-1}(x_{j+k}) = y_{j+k}$.

- We observe that $q \in \mathbb{P}_k$.
- It then follows by uniqueness of the interpolation polynomial that $q = p_{j,k}$.

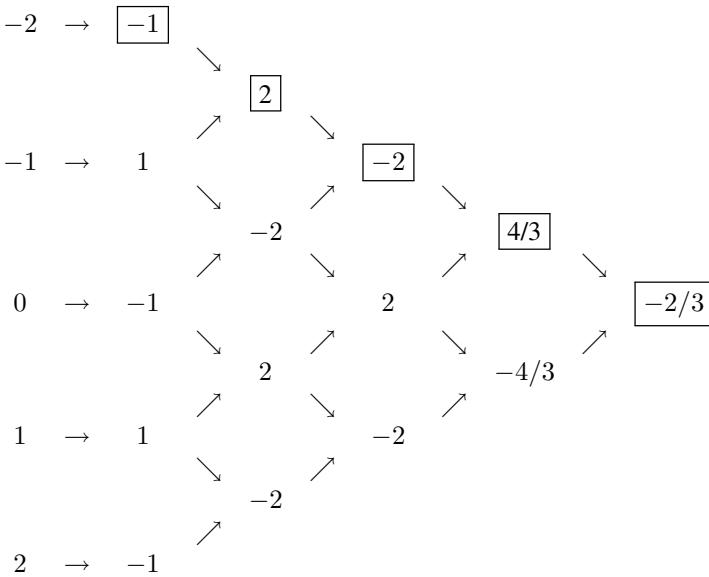
2. Let us compare the coefficient of t^k in the two parts of the equality (6.7). On the left, the coefficient is $[x_j, \dots, x_{j+k}]f$ using its definition. On the right, the power t^k appears when multiplying $\frac{-t}{x_{j+k} - x_j}$ by the t^{k-1} term in $p_{j,k-1}(t)$ (the coefficient is $[x_j, \dots, x_{j+k-1}]f$) and also by the multiplication of $\frac{t}{x_{j+k} - x_j}$ by t^{k-1} term in $p_{j+1,k-1}(t)$ (the coefficient is $[x_{j+1}, \dots, x_{j+k}]f$). We deduce that the common coefficient of t^k is

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

which is the formula.

¹⁵ a_k can vanish, for example when the sampled function is $f(t) = 1$ then $a_2 = \dots, a_n = 0$ and $1 = p_0 = p_1 = \dots = p_n$.

3. The first rows are the data. Using the previous formula, we deduce the computation beginning from the left by the $[x_j, x_{j+1}]$.



The Newton form of the interpolation polynomial is

$$p(t) = -1 + 2(t+2) - 2(t+2)(t+1) + \frac{4}{3}(t+2)(t+1)t - \frac{2}{3}(t+2)(t+1)t(t-1).$$

In the power basis $p(t) = (-2t^4 + 8t^2 - 3)/3$.

Exercise 6.6

1. For any real x , let $Q(x) = \prod_{j=1}^{n+1} \frac{(x-x_j)}{(t-x_j)}$.

Since $g(x) = f(x) - p(x) - e(t)Q(x)$, $f(x_i) = p(x_i)$ and $Q(x_i) = 0$, for $i = 1, \dots, n+1$, we deduce that $g(x_i) = 0$. Now, $e(t) = f(t) - p(t)$ and $Q(t) = 1$ so that $g(t) = 0$. Thus, g vanishes at $n+2$ distinct points. As soon as f admits a first derivative, g also does since p and Q are polynomials. We use Rolle Theorem to deduce that between two successive points where g vanishes, there exists a point where g' vanishes. Precisely, this new point is strictly between the previous ones. Hence, there exists a finite increasing sequence x_i^1 , $i = 1, \dots, n+1$ such that $g'(x_1^1) = g(x_2^1) = \dots = g(x_{n+1}^1) = 0$.

Michel Rolle (1652–1719) was a French mathematician. He is best known for Rolle's theorem (1691), and he deserves to be known as the co-inventor in Europe of Gaussian elimination (1690).



2. As soon as f admits a second derivative, from this sequence, we obtain a new sequence x_i^2 , $i = 1, \dots, n$ where g'' vanishes. By induction, there exists one

point $x_1^{n+1} = \xi$ such that $g^{(n+1)}(\xi) = 0$ whenever the $(n + 1)$ th derivative of f exists.

3. Let us differentiate g $n + 1$ times. Since p is a polynomial of degree at most n , we deduce that $p^{(n+1)} = 0$ and $Q^{(n+1)}(t) = \prod_{j=1}^{n+1} \frac{1}{(x - x_j)} \times (n + 1)!.$ Now $g^{(n+1)}(\xi) = 0$ implies $e(x) = \frac{1}{(n + 1)!} \prod_{j=1}^{n+1} (x - x_j) f^{(n+1)}(\xi).$

We notice that $e(t) = 0$ when $t = x_i$ for one $i = 1, \dots, n$ so in that case (6.1) is satisfied for any ξ .

Exercise 6.7

1. Since $f \in C^2([a, b])$, to each t in $[a, b]$ there exists c_1 depending on t such that

$$f(a) = f(t) + (a - t)f'(t) + (a - t)^2/2f''(c_1), \quad (6.17)$$

2. Similarly, there exists c_2 such that

$$f(b) = f(t) + (b - t)f'(t) + (b - t)^2/2f''(c_2). \quad (6.18)$$

3. We multiply (6.17) by $(b - t)$ and (6.18) by $(t - a)$ and add the two new equations. This gives

$$\begin{aligned} (b - t)f(a) + (t - a)f(b) &= (b - a)f(t) \\ &\quad + \frac{(t - a)(b - t)}{2} ((t - a)f''(c_1) + (b - t)f''(c_2)) \end{aligned}$$

from which we deduce

$$|(b - t)f(a) + (t - a)f(b) - (b - a)f(t)| \leq \frac{(t - a)(b - t)}{2}(b - a)M_2.$$

with $M_2 = \max_{a \leq z \leq b} |f''(z)|.$

4. The interpolating polynomial $p(t)$ is the unique polynomial in \mathbb{P}_1 that satisfies $p(a) = f(a)$ and $p(b) = f(b)$ so that $p(t) = \frac{(b - t)f(a) + (t - a)f(b)}{b - a}.$ From the previous inequality, we obtain

$$|p(t) - f(t)| \leq \frac{(t - a)(b - t)}{2}M_2.$$

We conclude since $(t - a)(b - t)$ is a polynomial of degree 2 which vanishes at a and b so that it has its maximum at the midpoint $(a + b)/2$ and this maximum has value $(b - a)^2/4.$

Exercise 6.8

```

tabn=[2,4,6,8,10,12,14,16,18,20];
a=-5;b=5;
tabl=zeros(length(tabn),4);
for i=1:length(tabn)
    n=tabn(i);
    h=(b-a)/n;
    x=a:h:b;
    y=1./(1+x.*x);
    coeff=polyfit(x,y,n);
    xnm=b-h/2;
    tabl(i,1)=n;
    tabl(i,2)=1/(1+xnm*xnm);
    tabl(i,3)=polyval(coeff,xnm);
    tabl(i,4)=abs(tabl(i,2)-tabl(i,3));
    t=a:0.01:b;
    pn=polyval(coeff,t);
    plot(x,y,'*',t,pn,t,1./(1+t.*t));
    title(['n = ',int2str(n)],'FontSize',16)
    pause
end;
tabl
disp('          n          f(t)          p(t)          error')

```

Comment: The Lagrange polynomial is computed by using the function `polyfit`; `polyfit(x,y,k)` is a least-squares function for $k < n$ and it gives the interpolant when $k = n$ where $n+1$ is the size of x . Precisely it gives the coefficients of the least-squares or interpolating polynomial in the power basis. ☐

6.5.3 Stable Lagrange Interpolation**Exercise 6.9**

1. We find

$$\begin{aligned}
 p(t) &:= \sum_{k=1}^{n+1} y_k \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{t - x_j}{x_k - x_j} = \ell(t) \sum_{k=1}^{n+1} y_k \frac{1}{t - x_k} \prod_{\substack{j=1 \\ j \neq k}}^{n+1} \frac{1}{x_k - x_j} \\
 &= \ell(t) \sum_{k=1}^{n+1} \frac{w_k}{t - x_k} y_k.
 \end{aligned}$$

2. Using the hint this follows by inserting $\ell(t) = 1 / \sum_{k=1}^{n+1} \frac{w_k}{t - x_k}$ in (6.11).
3. We find

$$w_k^m = \frac{1}{x_k - x_{m+1}} w_k^{m-1}, \quad k = 1, \dots, m, \quad w_{m+1}^m = \prod_{j=1}^m \frac{1}{x_{m+1} - x_j}.$$

Clearly this can be computed in $O(m)$ operations.

4a,4b. function lagweights.m

```
function W = lagweights(X)
r=length(X);
M=X*ones(1,r);
B=1./(eye(r)+M-M');
W=prod(B,2);
```

4c. function specialsum.m

```
function S=specialsum(X,Z,T)
[TT,XX]=meshgrid(T,X);
M=(diag(Z)*(1./(TT-XX)));
S=sum(M);
```

4d. lagpoint.m

```
X=pi*(0:1:100)/100;
X=-cos(X)';
Y=sqrt(abs(X));
W=lagweights(X);
T=-1:0.002:1;
S1=specialsum(X,Y.*W,T);
S2=specialsum(X,W,T);
P=S1./S2;
plot(X,Y,'--.b',T,P,'r')
```

Exercise 6.10

1. Since $\ell_p(x_i) = \delta_{p,i}$ and $\bar{\ell}_q(y_j) = \delta_{q,j}$ we obtain for all i, j

$$P(x_i, y_j) := \sum_{p=1}^{m+1} \sum_{q=1}^{n+1} z_{pq} \ell_p(x_i) \bar{\ell}_q(y_j) = z_{ij}.$$

2. Define

$$g_p(t) = \frac{\sum_{q=1}^{n+1} \frac{\bar{w}_q}{(t - y_q)} z_{pq}}{\sum_{q=1}^{n+1} \frac{\bar{w}_q}{t - y_q}}, \quad p = 1, 2, \dots, m+1.$$

By (6.12) we have $g_p(y_j) = z_{pj}$, for $p = 1, 2, \dots, m+1$. But then for all i, j

$$P(x_i, y_j) = \frac{\sum_{p=1}^{m+1} \frac{w_p}{(x_i - x_p)} g_p(y_j)}{\sum_{p=1}^{m+1} \frac{w_p}{x_i - x_p}} = z_{ij}.$$

3. The sampled function f.m

```
function z = f(s,t)
% $z = -20s \cdot (1-s) + 0.1t^3$ ;
z=sin(s+t);
end
```

4.

```
m=3;
n=7;
X=(0:1/m:1)'*pi;
Y=(0:1/n:1)'*2*pi;

[YY,XX]=meshgrid(Y,X);
ZZ=f(XX,YY);
U=(0:0.02:1)'*pi;
V=(0:0.02:1)'*2*pi;
W1=lagweights(X);
W2=lagweights(Y);
%numerатор
for p=1:length(X)
    S1(p,:)=specialsum(Y,W2.*ZZ(p,:),V);
end
for j=1:length(V)
    Suv(:,j)=specialsum(X,W1.*S1(:,j),U)';
end
%деноминатор
SX=specialsum(X,W1,U);
SY=specialsum(Y,W2,V);
[SSX,SSY]=meshgrid(SX,SY);
Slag=Suv ./ (SSX.*SSY)';
subplot(121)
[VV,UU]=meshgrid(V,U);
Sf=f(UU,VV);
surf(UU,VV, Sf)
title('The function')
subplot(122)
surf(XX,YY,ZZ)
title('Sampled Data')
figure
subplot(121)
surf(UU,VV, Slag)
title('Interpolating polynomial')
subplot(122)
surf(UU,VV, Slag-Sf)
title('Error')
```

6.5.4 Approximation of Derivatives, Part I

Exercise 6.11

1. Let $\ell(t) = \prod_{j=1}^{n+1} (t - x_j) = (t - x_j) \ell_j(t) / w_j$.

Suppose first $i \neq j$. Since $\ell(x_i) = 0$ for all i we find $\ell'_j(x_i) = \ell'(x_i) w_j / (x_i - x_j) = w_j / (w_i(x_i - x_j)) = m_{ij}$.

Consider next $i = j$. By the chain rule $\frac{d}{dt} \log \ell_j(t) = \ell'_j(t) / \ell_j(t)$, and since $\ell_j(x_j) = 1$, we obtain

$$\begin{aligned}\ell'_j(x_j) &= \frac{d}{dt} \log \ell_j(t)|_{t=x_j} = \frac{d}{dt} \sum_{\substack{k=1 \\ k \neq j}}^{n+1} (\log(t - x_k) - \log(x_j - x_k))|_{x=x_j} \\ &= \sum_{\substack{k=1 \\ k \neq j}}^{n+1} \frac{1}{x_j - x_k} = m_{jj}.\end{aligned}$$

2. We have $(Qe)_j = \sum_{\substack{k=1 \\ k \neq j}}^{n+1} \frac{1}{(x_j - x_k)} = m_{jj}$ for all j . Since the diagonal elements of $W^{-1}QW$ are zero this shows the result for $i = j$. When $i \neq j$ then $(Qe)_{i,j} = 0$ and $(W^{-1}QW)_{i,j} = m_{i,j}$.

3.

```
function M=mmatrix (x)
N=length (x);
w=lagweights (x);
Q=x*ones (1,N);
Q=1./ (eye (N)+Q-Q')-eye (N);
M=diag (1./w)*Q*diag (w)+diag (Q*ones (N,1));
```
4.

```
function dy=df(x,y)
dy=mmatrix (x)*(y (:));
```

Exercise 6.12

1.

```
function plotder (x,f,derf,m)
x=x (:);
N=length (x);
h=(x(N)-x(1))/m;
dy=df(x,f(x));
XX=x(1):h:x(N);
plot (XX,derf (XX),x,dy)
```

Exercise 6.13

1. We have $M * (1, 1, \dots, 1)^T = \mathbf{0}$ since the differentiation formula is exact for the constant function.

©Comment: The computation of the derivative with M is also exact (up to the rounding errors) for any data y sampled from polynomials of degree up to n . •

2. If \mathbf{y} is an approximate solution to the differential equation at x then $M\mathbf{y}$ is an approximation to the derivatives at x . But then $(M + I)\mathbf{y}$ is an approximation to $\mathbf{u}' + \mathbf{u}$ at x , and since $\mathbf{u}' + \mathbf{u} = 2e^x$ the solution of $(M + I)\mathbf{y} = \mathbf{b}$ will be an approximation to \mathbf{u} .

3.

```
function y=ode1(n)
x=0:1/n:1;
y=(mmatrix(x)+eye(length(x)))\ (2*exp(x(:)));
```

4. If $u(t) = 2 \sinh(t) + u_0 e^{-t}$ then $u'(t) = 2 \cosh(t) - u_0 e^{-t}$ and $u'(t) + u(t) = 2e^t$. Moreover, $u(0) = u_0$.

```
function e=ode2(n)
x=0:1/n:1;
y=ode1(n);
u=2*sinh(x)+y(1)*exp(-x);
e=max(abs(y(:)-u(:)));
```

Chapter 7

Bézier Curves and Bernstein Polynomials

This chapter gives an introduction to some mathematical tools used in Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), and Computer Aided Geometric Design (CAGD). Part of the mathematical objective is to find the equation for a curve, surface or solid, modeling a given physical form or image. Applications include modeling of car bodies, ship hulls and airplanes, medical imaging, animation films and video games.

In its simplest (and historical) form, one often uses Bézier curves which are parametric polynomials or piecewise polynomials, and where each polynomial piece is represented in terms of the Bernstein polynomials.¹ We also encountered the Bernstein polynomials briefly in Chap. 6.

Commercial systems uses B-splines or a rational version called Non Uniform Rational B-splines or NURBS. A geometric model constructed using B-splines is shown in Fig. 7.1. It is beyond the scope of this book to consider constructing such shapes and we restrict our study to Bézier curves, Bernstein polynomials, and the corresponding control polygon which mimics the shape of the curve and is useful for modeling.

For further study related to modeling we refer to the book [5].

A basic tool for dealing with Bézier curves is a recursive evaluation method called the de Casteljau algorithm.

In Exercise 7.1, we consider this algorithm and plot some curves and associated control polygons. In the following section, we study Bernstein polynomials, (Exercise 7.2) and their relationship to Bézier curves in Exercise 7.3. Exercise 7.4 gives an interesting opportunity to go back to linear algebra with the transformation from one polynomial basis to another.

We finish the chapter by applications: Exercise 7.5 about some shape properties and Exercises 7.7 and 7.6 about joining two or more Bézier curves.

¹ In the approximation theory, the term “Bernstein polynomial” is also used for an operator mapping a continuous function into a polynomial, see Chap. 11.

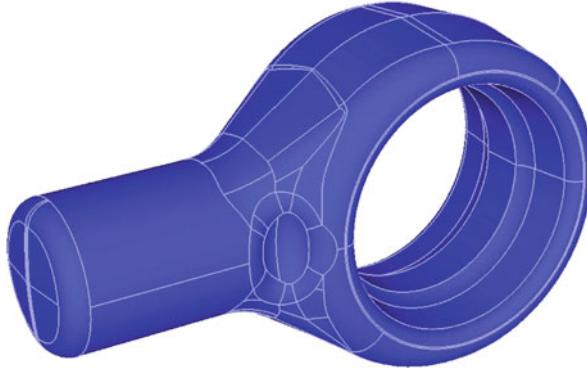


Fig. 7.1 A geometric model constructed using B-splines

In this chapter, the usual review is divided in two parts and is given at the beginning of the two first sections.

7.1 Bézier Curves and the de Casteljau Algorithm

Review: We will study the following recursive algorithm.

Algorithm 7.1 (de Casteljau Algorithm) Given $n + 1$ points $(\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n)$ in \mathbb{R}^d , $d \geq 2$, and a $t \in [0, 1]$. A point $\mathbf{P}_n(t)$ on a parametric curve $\mathbf{P}_n : [0, 1] \rightarrow \mathbb{R}^d$ is computed.

1. $\mathbf{b}_i^0(t) = \mathbf{b}_i$, $i = 0, \dots, n$
2. $\mathbf{b}_i^r(t) = (1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)$, $i = 0, \dots, n-r$, $r = 1, \dots, n$.
3. $\mathbf{P}_n(t) = \mathbf{b}_0^n(t)$

The computation of $\mathbf{b}_i^r = \mathbf{b}_i^r(t)$ for $n = 3$ and a given t is illustrated in Fig. 7.2. We start with the points $\mathbf{b}_i^0 = \mathbf{b}_i$, $i = 0, 1, 2, 3$, move left using the de Casteljau algorithm, and the final result is $\mathbf{b}_0^3 = \mathbf{P}_3(t)$.

It is often convenient to use the notation $\mathcal{B}(\mathbf{b}_0, \mathbf{b}_1, \dots, \mathbf{b}_n)$ for $\mathbf{P}_n : [0, 1] \rightarrow \mathbb{R}^d$. It is a plane curve for $d = 2$ and a space curve for $d = 3$. It is known as a **Bézier curve of degree n** . Values of d greater than 3 can be useful for constructing surfaces. The points $\{\mathbf{b}_0, \dots, \mathbf{b}_n\}$ are called **Bézier points**. They can be represented as a **Bézier point matrix** $\mathbf{C} := (\mathbf{b}_0, \dots, \mathbf{b}_n)^T$ with $N := n + 1$ rows and d columns. The polygon $\mathcal{C} = \mathcal{C}(\mathbf{b}_0, \dots, \mathbf{b}_n)$ with the Bézier points as vertices is called the **control polygon**.

It is often useful to define a Bézier curve over a parameter domain other than $[0, 1]$. We can define a Bézier curve $\mathbf{P} : [a, b] \rightarrow \mathbb{R}^d$ for an arbitrary interval $[a, b]$ by the formula

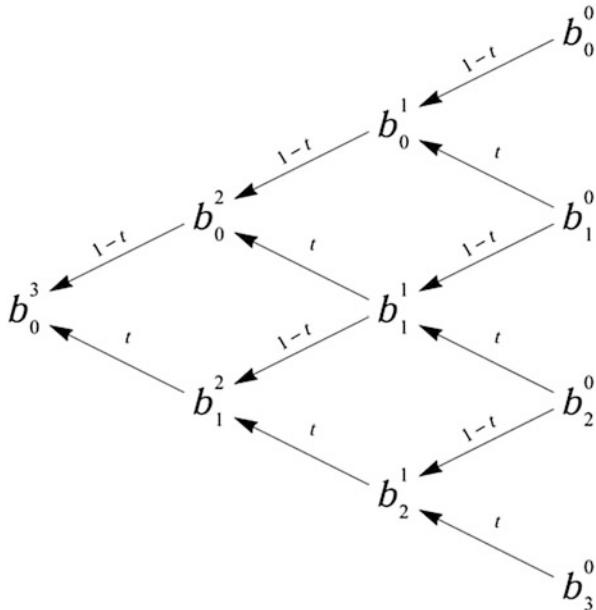


Fig. 7.2 The de Casteljau algorithm

$$\mathbf{P}(x) = \mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(t), \quad t := \frac{x - a}{b - a}. \quad (7.1)$$

For the MATLAB programs, we will write the components of the points \mathbf{b}_i in a row, for example, $\mathbf{b}_i = [x_i, y_i]$ or $\mathbf{b}_i = [x_i, y_i, z_i]$. \blacklozenge

Exercise 7.1. First properties of the Bézier curve

Pierre Étienne Bézier (1910–1999) was a French engineer at Renault, the car company, and one of the founders of the fields of solid, geometric and physical modeling as well as in the field of representing curves, especially in CAD/CAM systems. He became a leader in the transformation of design and manufacturing, through mathematics and computing tools, into computer-aided design and three-dimensional modeling.



1. Show that $\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(0) = \mathbf{b}_0$ and $\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(1) = \mathbf{b}_n$. \triangleright Math Hint² \triangleleft This means that the curve starts at \mathbf{b}_0 and arrives at \mathbf{b}_n and is known as the **end point property**.
2. Let $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}^d$ be an affine transformation (for example a translation, a symmetry, a rotation, a projection...), i.e., $\varphi((1-t)\mathbf{x} + t\mathbf{y}) = (1-t)\varphi(\mathbf{x}) + t\varphi(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ and $t \in \mathbb{R}$, and for $i = 0, \dots, n$ let us define $\tilde{\mathbf{b}}_i = \varphi(\mathbf{b}_i)$. Show that $\phi(\mathbf{b}_i^r(t)) = \tilde{\mathbf{b}}_i^r(t)$ for $i = 0, \dots, n-r$, and $r = 1, \dots, n$, where the

² Use an induction on r .

$\tilde{b}_i^r(t)$ are computed from the de Casteljau algorithm starting with the points \tilde{b}_i , $i = 0, 1, \dots, n$. ▷ *Math Hint*³ ◁

Thus, when $r = n$, $\varphi(\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)) = \mathcal{B}(\varphi(\mathbf{b}_0), \dots, \varphi(\mathbf{b}_n))$.

Paul de Casteljau (born 1930) is a French physicist and mathematician. In 1959, while working at Citroen, the car company, he developed an algorithm for evaluating calculations on a certain family of curves, which would later be formalized and popularized by engineer Pierre Bézier, and the curves called Bézier curves. De Casteljau's algorithm is widely used.



3. Write a function $b=bezier1(C, t)$ that given a Bézier point matrix C computes the value of the Bézier curve P_n at $t \in [0, 1]$. Test with $C = [0, 1, 2, 3; 1, 2, 2, 1]^T$ and $t = 0$, then $1/2$ and 1 .

```
>> C=[0,1,2,3;1,2,2,1]';  
>> b0=bezier1(C,0);  
>> b=bezier1(C,0.5);  
>> b1=bezier1(C,1);  
>> [b0;b;b1]  
ans =  
          0      1.0000  
    1.5000      1.7500  
    3.0000      1.0000
```

4. To plot a Bézier curve, we need to apply the de Casteljau algorithm for many values of t . Instead of calling `bezier1` many times with a loop, we will in this exercise modify the previous function so that values at the m t 's are computed simultaneously within the r loop. Write a function $B=bezier2(C, m)$ that given a Bézier point matrix C and a positive integer m computes the value of the Bézier curve P at m uniformly spaced values of t in $[0, 1]$. By using a 3 dimensional array b with $N = n + 1$ rows, d columns and m pages, we can do this using a m -loop for the initialization then only one r -loop. Initially page k of b contains a copy of C . At level r , row i of page k of b contains $b_{i-1}^r(t_k)$, $i = 1, \dots, N - r$, $k = 1, \dots, m$.

Test with $C = [0, 1, 2, 3; 1, 2, 2, 1]^T$ and $m = 5$. ▷ *MATLAB Hint*⁴ ◁

```
>> C=[0,1,2,3;1,2,2,1]';B=bezier2(C,5)  
B =  
          0      1.0000  
    0.7500      1.5625  
    1.5000      1.7500  
    2.2500      1.5625  
    3.0000      1.0000
```

³ Use an induction on r for all i .

⁴ The initial b can be computed initializing with $B = []$ then using m times $B = \text{cat}(3, B, C)$ while we can collect all t values in an array by $[, , T] = \text{ndgrid}(1:N, 1:d, 0:1/(m-1):1)$; To output B as a two dimensional array, use the function `squeeze`.

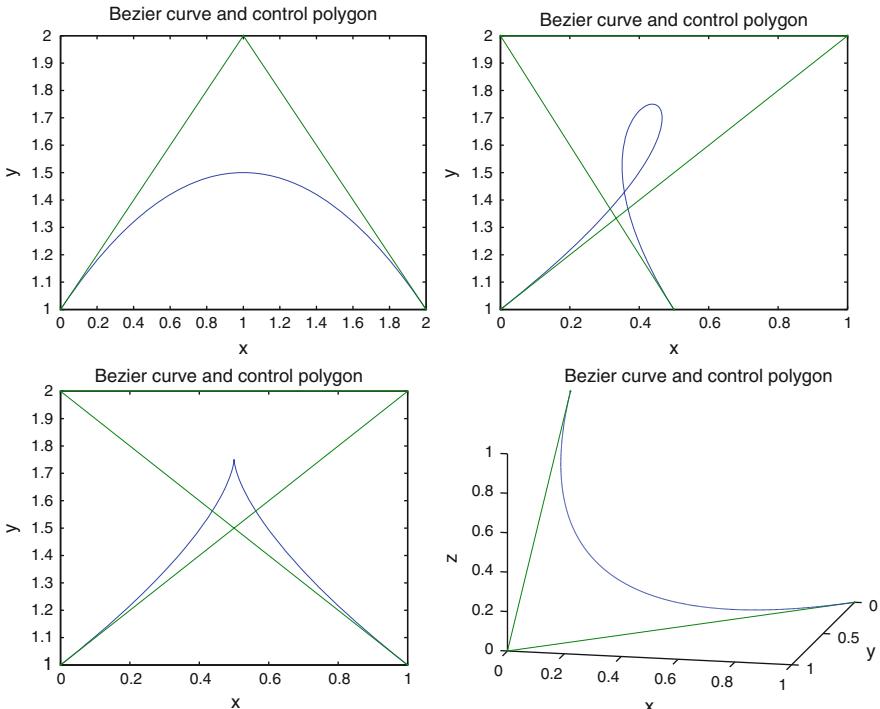


Fig. 7.3 Bézier curves. Clockwise from top: \mathbf{C}_1 , \mathbf{C}_2 , \mathbf{C}_3 , \mathbf{C}_4

Comment: Note that, given the uniform distribution of the abscissa of \mathbf{C} , the first component of $\mathbf{P}_n(t)$ is $x(t) = 3t$. This is proved using (7.8). •

5. Write a function `plotbezier(C, m, text)` that plots the control polygon and the curve using `bezier2` with m plot points and the functions `plot`, `plot3` based on the number of components d of the Bézier point matrix \mathbf{C} . `text` is a title for the plot. Make plots using

$\mathbf{C}_1 = [0, 1, 2; 1, 2, 1]^T$, $\mathbf{C}_2 = [0, 1, 0, 0.5; 1, 2, 2, 1]^T$, $\mathbf{C}_3 = [0, 1, 0, 1; 1, 2, 2, 1]^T$, then $\mathbf{C}_4 = [1, 0, 0; 0, 1, 0; 0, 0, 1]^T$, $m = 300$ and `text='Bezier curve and control polygon'`. Plots are in Fig. 7.3.

We will derive explicit parametric forms of these curves in the following section.

6. Write a program that selects a sequence of points, $[x, y]$ in the plane using the mouse. This sequence is the control polygon used to find and plot the corresponding Bézier curve. Save as `Bezieffrommouse`. < MATLAB Hint⁵ >

⁵ Use the function `ginput.m`.

7.2 Bernstein Polynomials

Review: For a positive integer n and $\binom{n}{i} = \frac{n!}{(n-i)!i!}$, the usual binomial coefficients, we define the **Bernstein polynomials** by

$$\begin{aligned} B_i^n(t) &= \binom{n}{i} t^i (1-t)^{n-i}, \text{ for } i = 0, \dots, n \\ B_i^n(t) &= 0, \text{ for } i \notin \{0, \dots, n\} \end{aligned}$$

Sergei Natanovich Bernstein (1880–1968) was a Russian and Soviet mathematician known for his contributions to partial differential equations, differential geometry, probability theory, and approximation theory. In his thesis (1904 in Paris), he solved Hilbert's nineteenth problem on the analytic solution of elliptic differential equations. He studied the connection between smoothness properties of a function and its approximations by polynomials.



Clearly, for $i = 0, \dots, n$ and $t \in (0, 1)$, $B_i^n(t) > 0$. We first show that the Bernstein polynomials for $i = 0, \dots, n$, constitute a basis for \mathbb{P}_n and some useful properties of these polynomials.

Then we will show the close link between a Bézier curve $\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)$ and Bernstein polynomials

$$\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(t) = \mathbf{b}_0^n(t) = \sum_{j=0}^n \mathbf{b}_j B_j^n(t). \quad (7.2)$$

Next we will prove that the derivatives of a Bézier curve are also Bézier curves that can be expressed taking forward differences of the Bézier coefficients. We will show that

$$\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)^{(\ell)}(t) = \frac{n!}{(n-\ell)!} \sum_{i=0}^{n-\ell} \Delta^\ell \mathbf{b}_i B_i^{n-\ell}(t), \quad \ell = 0, 1, \dots, n \quad (7.3)$$

$$= \mathcal{B} \left(\frac{n!}{(n-\ell)!} \Delta^\ell \mathbf{b}_0, \dots, \frac{n!}{(n-\ell)!} \Delta^\ell \mathbf{b}_{n-\ell} \right) (t). \quad (7.4)$$

where $\Delta^0 \mathbf{b}_i := \mathbf{b}_i$ and for $k \geq 1$

$$\Delta^\ell \mathbf{b}_i := \Delta^{\ell-1} \mathbf{b}_{i+1} - \Delta^{\ell-1} \mathbf{b}_i = \sum_{j=0}^{\ell} (-1)^{\ell-j} \binom{p}{j} \mathbf{b}_{i+j}.$$



Exercise 7.2. Elementary properties

1. Show that

$$(xt + 1 - t)^n = \sum_{i=0}^n x^i B_i^n(t), \quad t, x \in \mathbb{R}, \quad n \geq 0. \quad (7.5)$$

2. Deduce that for $t \in \mathbb{R}$,

$$1 = \sum_{i=0}^n B_i^n(t), \quad n \geq 0, \quad (7.6)$$

$$t = \sum_{i=0}^n \frac{i}{n} B_i^n(t), \quad n \geq 1, \quad (7.7)$$

$$t^k = \sum_{i=k}^n \frac{\binom{i}{k}}{\binom{n}{k}} B_i^n(t), \quad 0 \leq k \leq n. \quad (7.8)$$

▷ *Math Hint*⁶ ◁

3. Show that the set of Bernstein polynomials of degree n , $\{B_i^n(t)\}_{i=0,\dots,n}$ is a basis for \mathbb{P}_n .
 4. Show

- a. The recurrence relation

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \quad n \geq 1, \quad i \in \mathbb{Z}, \quad t \in \mathbb{R}. \quad (7.9)$$

▷ *Math Hint*⁷ ◁

- b. The differentiation formula

$$\frac{dB_i^n}{dt}(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)), \quad n \geq 1, \quad i \in \mathbb{Z}, \quad t \in \mathbb{R}. \quad (7.10)$$

The computation of $B_i^n(t)$ for $n \leq 3$ and a fixed t is illustrated in Fig. 7.4. We start with the degree zero polynomial $B_0^0(t) = 1$ and move right using the recurrence relation (7.9). Note that the labels are the same as in Fig. 7.2, we have just reversed the direction of the arrows.

Exercise 7.3. Bernstein polynomials and Bézier curves

1. For $r = 0, \dots, n$ and $i = 0, \dots, n-r$, show that

$$\mathbf{b}_i^r(t) = \sum_{j=0}^r \mathbf{b}_{i+j} B_j^r(t), \quad t \in [0, 1], \quad (7.11)$$

▷ *Math Hint*⁸ ◁

2. Deduce (7.2).
 3. To convert (7.10) into a differentiation formula for Bézier curves, let $u_k \in \mathbb{R}^i$ and $v_k \in \mathbb{R}^j$ for $k = 0, 1, \dots, m+1$ and $i, j \in \mathbb{N}$, with $i = 1$ or $j = 1$ and recall

⁶ Take derivatives with respect to x then set $t = 1$.

⁷ Distinguish different cases of i .

⁸ Induction on r .

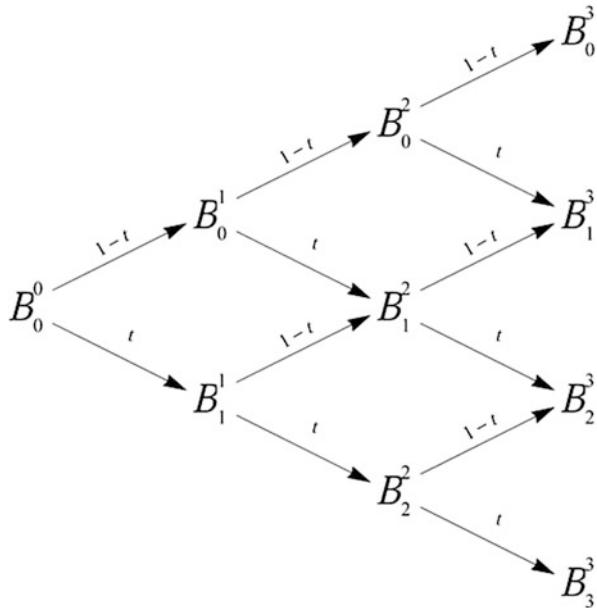


Fig. 7.4 The Bernstein basis evaluation

that $\Delta v_k := v_{k+1} - v_k$ is the forward difference. Show the summation by parts formula

$$\sum_{k=0}^m u_k \Delta v_k = u_{m+1} v_{m+1} - u_0 v_0 - \sum_{k=0}^m v_{k+1} \Delta u_k. \quad (7.12)$$

▷ *Math Hint*⁹ ◁

4. Deduce (7.3). ▷ *Math Hint*¹⁰ ◁

5. Prove that

$$\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)'(0) = n(\mathbf{b}_1 - \mathbf{b}_0), \quad \mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)'(1) = n(\mathbf{b}_n - \mathbf{b}_{n-1}). \quad (7.13)$$

This is known as the **tangent end point property** and shows that the control polygon is tangent to the Bézier curve at the endpoints.

Exercise 7.4. Power basis and Bernstein basis for Bézier curves

Let $\mathbf{P}(t) = \sum_{i=0}^n b_i B_i^n(t)$ be a Bézier curve. In this exercise we determine coefficients $\mathbf{a}_0, \dots, \mathbf{a}_n$ so that $\mathbf{P}(t) = \sum_{k=0}^n \mathbf{a}_k t^k$.

⁹ Prove $u_m \Delta v_m = u_{m+1} v_{m+1} - u_m v_m - v_{m+1} \Delta u_m$.

¹⁰ Use induction, (7.10), and (7.12).

1. Show using the differentiation formula (7.3) that

$$\mathbf{a}_k = \binom{n}{k} \Delta^k \mathbf{b}_0, \quad k = 0, 1, \dots, n. \quad (7.14)$$

▷ *Math Hint*¹¹ ◁

2. Write a program function `a=berNSTINTOPower(b)` that computes the coefficients in (7.14). Test it on the coefficients \mathbf{C}_3 in Exercise 7.1. Show that for \mathbf{C}_3 , we have $\mathbf{P}'_3(1/2) = \mathbf{0}$ indicating the cusp in Fig. 7.3.

```
>> a=berNSTINTOPower(C3)
a =
    0      1
    3      3
   -6     -3
    4      0
```

3. In the following, we convert a parametric polynomial plane or space curve of degree n in the form $\mathbf{P}(t) = \sum_{k=0}^n \mathbf{a}_k t^k$ to a Bézier curve $\mathbf{P}(t) = \sum_{i=0}^n \mathbf{b}_i B_i^n(t)$. Show that

$$\begin{bmatrix} \mathbf{b}_0 \\ \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} = \begin{bmatrix} \binom{0}{0} & & & & \\ \binom{1}{0} & \binom{1}{1} & & & \\ \binom{2}{0} & \binom{2}{1} & \binom{2}{2} & & \\ \vdots & \vdots & \vdots & \ddots & \\ \binom{n}{0} & \binom{n}{1} & \binom{n}{2} & \cdots & \binom{n}{n} \end{bmatrix} \begin{bmatrix} d_0 & & & & \\ & d_1 & & & \\ & & d_2 & & \\ & & & \ddots & \\ & & & & d_n \end{bmatrix} \begin{bmatrix} \mathbf{a}_0 \\ \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}, \quad (7.15)$$

where $d_k := \binom{n}{k}^{-1}$, $k = 0, 1, \dots, n$. ▷ *Math Hint*¹² ◁

© **Comment:** The operation needs the “coefficients” \mathbf{a}_i and \mathbf{b}_i to be written as rows of length d . ☺

4. Write a program function `a=powertobernstein(b)` that computes the coefficients in (7.15). Test it on the coefficients computed from \mathbf{C}_3 in Exercise 7.4.

```
>> b=powertobernstein(berNSTINTOPower(C3))
b =
    0      1
    1      2
    0      2
    1      1
```

¹¹ Use the Taylor expansion of $\mathbf{P}(t)$.

¹² Use (7.8).

7.3 Shape Preservation

In this section we will consider Bézier curves defined from scalar y -values. For $i = 0, \dots, n$, if $b_i \in \mathbb{R}$, consider the Bézier curve $\mathbf{P}_n(t) = \mathcal{B}(b_0, \dots, b_n)(t) = (x(t), y(t))$ with

$$\mathbf{b}_i = \left(\frac{i}{n}, b_i \right), \quad i = 0, 1, \dots, n. \quad (7.16)$$

The theme of this section is that the control polygon models the corresponding Bézier curve.

Exercise 7.5. Consider for $n \geq 1$ the Bézier curve $\mathbf{P}(t) = (x(t), y(t))$ defined from the Bézier points given by (7.16). Recall that

$$\Delta^k b_i = \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} b_{i+j} \text{ for } i = 0, 1, \dots, n-k, k = 0, 1, \dots, n.$$

1. Show that $x(t) = t$. \triangleright Math Hint¹³ \triangleleft
2. Show that if $\Delta^k b_i \geq 0$ for $i = 0, 1, \dots, n-k$ then $y^{(k)}(t) \geq 0$ for $t \in [0, 1]$ and any $k = 0, 1, \dots, n$. \triangleright Math Hint¹⁴ \triangleleft
3. Suppose $b_i = \exp(i/n)$ for $i = 0, 1, \dots, n$. Show that $y^{(k)}(t) > 0$ for $t \in [0, 1]$ and $k = 0, 1, \dots, n$.
4. For $n = 5$ make a plot of the Bézier curve in 3., together with its control polygon and the exponential function.
5. The conditions just given are sufficient for nonnegativity of the k th derivative of y , but they are not necessary.
For $k = 0$ consider the parametric curve $\mathbf{P} = (x, y) : [0, 1] \rightarrow \mathbb{R}^2$ given by $\mathbf{P}(t) = (x(t), y(t)) = (t, 1 - 3t + 9t^3)$. Show that $\mathbf{P}(t) \geq 1/3$ for $t \in [0, 1]$. Find the Bernstein basis coefficients (b_0, \dots, b_3) of y . \triangleright Math Hint¹⁵ \triangleleft
6. Check the answer using the program `powertobernstein` (cf. Exercise 7.4). Make a plot of the curve and the control polygon and observe that y is positive, but b_2 is negative.
7. In this problem we study a Bézier curve (x, y) with y monotonically decreasing, but y does not have monotonically decreasing Bernstein basis coefficients. Consider the parametric curve $\mathbf{Q} = (x, y) : [0, 1] \rightarrow \mathbb{R}^2$ given by $\mathbf{Q}(t) = (x(t), y(t)) = (t, 2 - 3t + 6t^2 - 4t^3)$. Show that y is monotonically decreasing on $[0, 1]$. Make a plot of the curve and the control polygon as in the previous exercise.

¹³ Use (7.7).

¹⁴ Use (7.3).

¹⁵ Use (7.15).

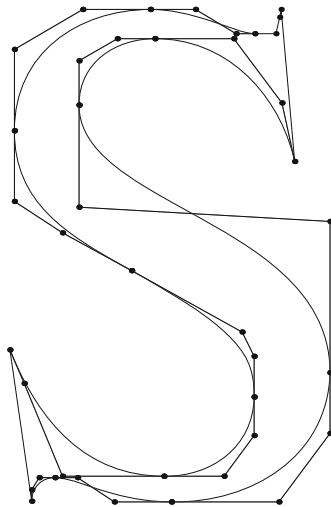


Fig. 7.5 A Postscript Times Roman font as a piecewise cubic Bézier curve

7.4 Smooth Joining of Bézier Curves

Pieces of Bézier curves can be joined together to form a smooth composite curve. Postscript fonts were designed this way. An example is shown in Fig. 7.5. Here cubic Bézier curves are joined with smoothness C^1 except at the top right- and bottom left segments that are joined with smoothness C^0 .

Exercise 7.6. Joining Elementary Bézier curves

In the plane, we consider the Bézier curve $\mathbf{P}_1(x) = \mathcal{B}_1(\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)(x)$ for $x \in [0, 1]$ where $\mathbf{a}_0 = [-1, 0]$, $\mathbf{a}_1 = [-1, 1]$, $\mathbf{a}_2 = [1, 0]$, $\mathbf{a}_3 = [1, 1]$.

1. What is the degree n of $\mathbf{P}_1(x)$?
2. We want to join the curve $\mathbf{P}_1(x)$ to the Bézier curve $\mathbf{P}_2(x)$, $x \in [1, 2]$ with a C^1 junction at $\mathbf{b}_0 = \mathbf{a}_3$. $\mathbf{P}_2(x) = \mathcal{B}_1(\mathbf{b}_0, \mathbf{b}_1, \mathbf{b}_2)(x - 1)$ and $\mathbf{b}_2 = [3, 0]$ is given. Use $\mathbf{P}_1(1) = \mathbf{P}_2(1)$ to find \mathbf{b}_1 . ▷ Math Hint¹⁶ ◁
3. Now, we want to construct a Bézier curve $\mathbf{P}_3(x)$, $x \in [2, 3]$ $\mathbf{P}_3(x) = \mathcal{B}_1(\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)(x - 2)$ from $\mathbf{P}_2(x)$ to $\mathbf{P}_1(x)$ with $\mathbf{c}_0 = \mathbf{b}_2$ and a C^2 junction at $\mathbf{c}_3 = \mathbf{a}_0$. Find the points \mathbf{c}_1 and \mathbf{c}_2 . ▷ Math Hint¹⁷ ◁
4. Plot the three control polygons and corresponding curves using the program `bezier2` of Exercise 7.1. Save as `joinbezier`. ▷ MATLAB Hint¹⁸ ◁

¹⁶ Use (7.3).

¹⁷ Write two equations for the C^2 junction.

¹⁸ Or use the function `plotbezier` with the command `hold on`.

Exercise 7.7. Generalization

Let $a < b < c$, $m, n \geq 0$, $\mathbf{d}_0, \dots, \mathbf{d}_m \in \mathbb{R}^d$ and $\mathbf{e}_0, \dots, \mathbf{e}_n \in \mathbb{R}^d$. Consider the plane curve $\mathbf{P} = (x(t), y(t)) : [a, c] \rightarrow \mathbb{R}^2$ given by

$$\mathbf{P}(x) = \begin{cases} \mathbf{P}_1(x) = \mathcal{B}_1(\mathbf{d}_0, \dots, \mathbf{d}_m) \left(\frac{x-a}{b-a} \right) & \text{if } a \leq x < b, \\ \mathbf{P}_2(x) = \mathcal{B}_2(\mathbf{e}_0, \dots, \mathbf{e}_n) \left(\frac{x-b}{c-b} \right) & \text{if } b \leq x \leq c. \end{cases}$$

1. Show that \mathbf{P} is continuous at $x = b$ if $\mathbf{d}_m = \mathbf{e}_0$.
2. If $m, n \geq 1$, $\mathbf{d}_m = \mathbf{e}_0$, and $\frac{m}{b-a}(\mathbf{d}_m - \mathbf{d}_{m-1}) = \frac{n}{c-b}(\mathbf{e}_1 - \mathbf{e}_0)$ then show that \mathbf{P} is continuous and has a continuous derivative at $x = b$. If $\mathbf{d}_m - \mathbf{d}_{m-1} = \mathbf{e}_1 - \mathbf{e}_0$ then \mathbf{P} has a continuous tangent at $x = b$.
3. If $m = n \geq r$, $c - b = b - a$, and $\Delta^k \mathbf{d}_{n-k} = \Delta^k \mathbf{e}_0$ for $k = 0, 1, \dots, r$, then $\mathbf{P}^{(k)}$ is continuous at $x = b$ for $0 \leq k \leq r$.
4. Make a plot of $\mathbf{P} : [-1, 1] \rightarrow \mathbb{R}^2$ in the case where $\mathbf{P}_1(x) = (x - 1, 2 - 3x + 6x^2 - 4x^3)$ and $\mathbf{P}_2 = (x, 1 - 3x + 9t^x)$ and $b = 0$, (cf. Exercise 7.5).
5. Show that \mathbf{P} is continuous and has a continuous first derivative at $x = 0$.

7.5 Solutions

7.5.1 Bézier Curves and the de Casteljau Algorithm

Exercise 7.1:

1. By induction, at each step r from 0 to n , $\mathbf{b}_0^r(0) = \mathbf{b}_0$ so that $\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(0) = \mathbf{b}_0^n(0) = \mathbf{b}_0$ and similarly $\mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(1) = \mathbf{b}_0^n(1) = \mathbf{b}_n$.
2. For $t \in [0, 1]$, by Algorithm 7.1, $\tilde{\mathbf{b}}_i^0(t) = \tilde{\mathbf{b}}_i = \varphi(\mathbf{b}_i) = \varphi(\mathbf{b}_i^0(t))$. Next, $\tilde{\mathbf{b}}_i^{r-1}(t) = \varphi(\mathbf{b}_i^{r-1}(t))$ for $0 < r \leq n$, all i and all t . Then, with 2. in Algorithm 7.1,

$$\begin{aligned} \tilde{\mathbf{b}}_i^r(t) &= (1-t)\tilde{\mathbf{b}}_i^{r-1}(t) + t\tilde{\mathbf{b}}_{i+1}^{r-1}(t) \\ &= (1-t)\varphi(\mathbf{b}_i^{r-1}(t)) + t\varphi(\mathbf{b}_{i+1}^{r-1}(t)) \\ &= \varphi((1-t)\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t)) = \varphi(\mathbf{b}_i^r(t)). \end{aligned}$$

By induction $\varphi(\mathbf{b}(t))$ is a Bézier curve with control polygon $\mathcal{C}(\varphi(\mathbf{b}_0), \dots, \varphi(\mathbf{b}_n))$.

3. Function `bezier1.m`

```
function b=bezier1(C, t)
%evaluate at one point, control polygon in R^d d>=2,
N=size(C, 1);
for r=N-1:-1:1
```

```

C(1:r,:)=(1-t)*C(1:r,:)+t*C(2:r+1,:);
end
b=C(1,:);

```

4. Function `bezier2.m`

```

function b=bezier2(C,m)
[N,d]=size(C); %N=n+1
%initialization
B=[];
for k=1:m
    B=cat(3,B,C);
end
t=0:1/(m-1):1;
[~,~,TT]=ndgrid(1:N,1:d,t);
%de Casteljau
for r=N-1:-1:1
    B(1:r,:,:)=(1-TT(1:r,:,:)).*B(1:r,:,:)+...
        TT(1:r,:,:).*B(2:r+1,:,:);
end
b=squeeze(B(1,:,:));

```

5. Function `plotbezier.m`

```

function plotbezier(C,m, text)
d=size(C,2);
B=bezier2(C,m);
if d==2
    plot(B(:,1),B(:,2),C(:,1),C(:,2),'.-')
else
    plot3(B(:,1),B(:,2),B(3,:),C(:,1),C(:,2),...
        C(:,3),'.-')
    zlabel('z','Fontsize',14)
end
xlabel('x','Fontsize',14)
ylabel('y','Fontsize',14)
title(text,'Fontsize',14)

```

6. Function `bezierfrommouse.m`

```

axis([0 10 0 10])
hold on
axis off
b = [];
i = 0;
% Loop, picking up the points.
LW='Left mouse button picks points, '
RW=' Right mouse button picks last point.'
title([LW,RW])
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'rx')
    i = i+1;

```

```

b(i,:) = [xi, yi];
end
b
plotbezier(b,500,'Bezier Curve and control polygon')

```

7.5.2 Bernstein Basis Polynomials

Exercise 7.2:

1. $(xt + 1 - t)^n = \sum_{i=0}^n \binom{n}{i} (xt)^i (1-t)^{n-i} = \sum_{i=1}^n x^i B_i^n(t)$ by the binomial expansion of $(xt + 1 - t)^n$ for any $x, t \in \mathbb{R}$.
2. • We set $x = 1$ in (7.5).
 - Differentiate both sides of (7.5) with respect to x and set $x = 1$.
 - Differentiate both sides of (7.5) k times with respect to x

$$\frac{n!}{(n-k)!} t^k (xt + 1 - t)^{n-k} = \sum_{i=k}^n \frac{i!}{(i-k)!} x^{i-k} B_i^n(t).$$

Set $x = 1$ and rearrange

$$t^k = \sum_{i=k}^n \frac{(n-k)! i!}{n! (i-k)!} B_i^n(t) = \sum_{i=k}^n \frac{\binom{i}{k}}{\binom{n}{k}} B_i^n(t).$$

which is (7.8).

3. We have shown that each power basis element t^k can be written as a linear combination of the $n+1$ Bernstein polynomials of degree n . Thus the space spanned by these polynomials contains \mathbb{P}_n . Since \mathbb{P}_n has dimension $n+1$ the two spaces must be equal and the Bernstein basis polynomials must be linearly independent.
4. a. • For $i \notin \{0, \dots, n\}$, $B_i^n(t) = 0 = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$.
 - For $i = 0$, $B_0^n(t) = (1-t)^n = (1-t)B_0^{n-1}(t) = (1-t)B_0^{n-1}(t) + tB_{-1}^{n-1}(t)$ as $B_{-1}^{n-1}(t) = 0$ and similarly
For $i = n$, $B_n^n(t) = (1-t)B_n^{n-1}(t) + tB_{n-1}^{n-1}(t)$.
 - For $i \in \{1, \dots, n-1\}$,

$$\begin{aligned}
 & (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) \\
 &= (1-t) \binom{n-1}{i} t^i (1-t)^{n-1-i} + t \binom{n-1}{i-1} t^{i-1} (1-t)^{n-1-i+1} \\
 &= \left(\binom{n-1}{i} + \binom{n-1}{i-1} \right) t^i (1-t)^{n-i} \\
 &= \binom{n}{i} t^i (1-t)^{n-i} = B_i^n(t).
 \end{aligned}$$

- b. • For $i \notin \{0, \dots, n\}$, both sides of (7.10) are zero.
• For $i = 0$ and $i = n$, since $B_{-1}^{n-1}(t) = B_n^{n-1}(t) = 0$

$$\frac{dB_0^n}{dt}(t) = \frac{d}{dt}(1-t)^n = -n(1-t)^{n-1} = n(B_{-1}^{n-1}(t) - B_0^{n-1}(t)),$$

$$\frac{dB_n^n}{dt}(t) = \frac{d}{dt}t^n = nt^{n-1} = n(B_{n-1}^{n-1}(t) - B_n^{n-1}(t)).$$

- For $i \in \{1, \dots, n-1\}$. Since $i \binom{n}{i} = n \binom{n-1}{i-1}$ and $(n-i) \binom{n}{i} = n \binom{n-1}{i}$ we find

$$\begin{aligned}\frac{dB_i^n}{dt}(t) &= i \binom{n}{i} t^{i-1} (1-t)^{n-i} - (n-i) \binom{n}{i} t^i (1-t)^{n-i-1} \\ &= n \binom{n-1}{i-1} t^{i-1} (1-t)^{n-i} - n \binom{n-1}{i} t^i (1-t)^{n-i-1} \\ &= n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))\end{aligned}$$

Exercise 7.3:

1. We show (7.11) by induction.

For $r = 0$, by definition, $\mathbf{b}_i^0 = b_i = \sum_{j=0}^0 \mathbf{b}_{i+j} B_j^0(t)$.

Let us assume that for $r \geq 1$, $\mathbf{b}_i^{r-1}(t) = \sum_{j=0}^{r-1} \mathbf{b}_{i+j} B_j^{r-1}(t)$ for $i = 0, \dots, n-r+1$. Then using the de Casteljau Algorithm 7.1,

$$\begin{aligned}\mathbf{b}_i^r(t) &= (1-t) \sum_{j=0}^{r-1} \mathbf{b}_{i+j} B_j^{r-1}(t) + t \sum_{j=0}^{r-1} \mathbf{b}_{i+1+j} B_j^{r-1}(t) \\ &= \mathbf{b}_i(1-t) B_0^{r-1}(t) + \sum_{k=1}^{r-1} \mathbf{b}_{i+k} ((1-t) B_k^{r-1}(t) + t B_{k-1}^{r-1}(t)) \\ &\quad + \mathbf{b}_{i+r} t B_{r-1}^{r-1}(t) \\ &= \mathbf{b}_i B_0^r(t) + \sum_{k=1}^{r-1} \mathbf{b}_{i+k} B_k^r(t) + \mathbf{b}_{i+r} B_r^r(t) = \sum_{k=0}^r \mathbf{b}_{i+k} B_k^r(t).\end{aligned}$$

2. Equation (7.2) is obtained in the case $r = n$.

3. Let us prove (7.12) by induction on m . To begin the proof of (7.12), we notice that for any m

$$u_m \Delta v_m = u_{m+1} v_{m+1} - u_m v_m - v_{m+1} \Delta u_m. \quad (7.17)$$

which is obtained by

$$\begin{aligned}u_m \Delta v_m &= u_m v_{m+1} - u_m v_m + u_{m+1} v_{m+1} - u_{m+1} v_{m+1} \\ &= u_{m+1} v_{m+1} - u_m v_m - v_{m+1} \Delta u_m.\end{aligned}$$

From (7.17), we obtain (7.12) for $m = 0$.

Suppose (7.12) holds for $m - 1$. Then using (7.17),

$$\begin{aligned} \sum_{k=0}^m u_k \Delta v_k &= \sum_{k=0}^{m-1} u_k \Delta v_k + u_m \Delta v_m = u_m v_m - u_0 v_0 - \sum_{k=0}^{m-1} v_{k+1} \Delta u_k \\ &\quad + u_{m+1} v_{m+1} - u_m v_m - v_{m+1} \Delta u_m \\ &= u_{m+1} v_{m+1} - u_0 v_0 - \sum_{k=0}^m v_{k+1} \Delta u_k, \end{aligned}$$

which finishes the proof.

4. Let $\mathbf{b}(t) = \mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(t)$. We use induction again.

For $\ell = 0$, then (7.3) is (7.2).

From ℓ to $\ell + 1$, we use (7.10) with n replaced by $n - \ell$, (7.12) with $u_k = \Delta^\ell \mathbf{b}_k$ with $m = n - \ell$ and $v_k = B_{k-1}^{n-\ell-1}(t)$. Then

$$\begin{aligned} \frac{(n - \ell - 1)!}{n!} \frac{d^{\ell+1} \mathbf{b}}{dt^{\ell+1}}(t) &= \frac{1}{n - \ell} \sum_{k=0}^{n-\ell} \Delta^\ell \mathbf{b}_k \frac{d}{dt} B_k^{n-\ell}(t) \\ &= \frac{1}{n - \ell} \sum_{k=0}^{n-\ell} \Delta^\ell \mathbf{b}_k [-(n - \ell) \Delta B_{k-1}^{n-\ell-1}(t)] \\ &= - \sum_{k=0}^{n-\ell} u_k \Delta v_k \\ &= -u_{n-\ell+1} v_{n-\ell+1} + u_0 v_0 + \sum_{k=0}^{n-\ell} v_{k+1} \Delta u_k \\ &= \sum_{k=0}^{n-\ell-1} \Delta^{\ell+1} \mathbf{b}_k B_k^{n-\ell-1}(t). \end{aligned}$$

The last equality follows since

$$v_0 = B_{-1}^{n-\ell-1}(t) = 0 \text{ and } v_{n-\ell+1} = B_{n-\ell}^{n-\ell-1}(t) = 0.$$

5. Again, we fix $\mathbf{b}(t) = \mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)(t)$. Using the differentiation formula (7.3) we find $\mathbf{b}'(t) = n \sum_{i=0}^{n-1} \Delta \mathbf{b}_i B_i^{n-1}(t)$. By the end point property for Bézier curves we obtain $\mathbf{b}'(0) = n \Delta \mathbf{b}_0 = n(\mathbf{b}_1 - \mathbf{b}_0)$ and $\mathbf{b}'(1) = n \Delta \mathbf{b}_{n-1} = n(\mathbf{b}_n - \mathbf{b}_{n-1})$.

Exercise 7.4:

1. Let $\mathbf{P} = \mathcal{B}(\mathbf{b}_0, \dots, \mathbf{b}_n)$. Since $\mathbf{P}(t) = \sum_{k=0}^n \frac{\mathbf{P}^{(k)}(0)}{k!} t^k$ we find from the differentiation formula

$$\mathbf{a}_k = \frac{\mathbf{P}^{(k)}(0)}{k!} = \frac{n!}{k!(n-k)!} \sum_{i=0}^{n-k} \Delta^k \mathbf{b}_i B_i^{n-k}(0) = \binom{n}{k} \Delta^k \mathbf{b}_0.$$

2.

```
function a=berNSTEINTOPOWER(b)
[N,~]=size(b);
bi=zeros(N+1,1); bi(2)=1;
for k=2:N
    b(k:N,:)=b(k:N,:)-b(k-1:N-1,:);
    bi(2:k+1)=bi(1:k)+bi(2:k+1);
end
a=diag(bi(2:N+1))*b;
```

3. We want to show that

$$\begin{bmatrix} \mathbf{b}_0 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} = \mathbf{U} \mathbf{D} \begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}, \quad (7.18)$$

where $\mathbf{D} = \text{diag}(d_0, \dots, d_n)$ and $\mathbf{U} = (u_{i,k})_{i,k=0}^n$ are the matrices in (7.15). Note that $u_{i,k} = 0$ for $i < k$ and $\binom{i}{k}$ otherwise. By (7.8)

$$t^k = \sum_{i=k}^n \frac{\binom{i}{k}}{\binom{n}{k}} B_i^n(t) = \sum_{i=0}^n d_k u_{i,k} B_i^n(t) = [B_0^n(t) \ \dots \ B_n^n(t)] \mathbf{U}(:, k) d_k.$$

But then

$$\mathbf{P}(t) = [t^0 \ \dots \ t^n] \begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix} = [B_0^n(t) \ \dots \ B_n^n(t)] \mathbf{U} \mathbf{D} \begin{bmatrix} \mathbf{a}_0 \\ \vdots \\ \mathbf{a}_n \end{bmatrix}.$$

Now (7.18) follows by linear independence of the Bernstein basis polynomials.

4.

```
function b=powERTOBERSNSTEIN(a)
N=size(a,1);
Pa=abs(pascal(N,1));
D=diag(1./Pa(N,:));
b=Pa*D*a;
```

7.5.3 Shape Preservation

Exercise 7.5

- We have $\mathbf{P} = \sum_{i=0}^n (\frac{i}{n}, b_i) B_i^n$. By Exercise 7.2, $x(t) = \sum_{i=0}^n \frac{i}{n} B_i^n(t) = t$.
- Since the Bernstein basis polynomials are nonnegative on $[0, 1]$, it follows from the differentiation formula (7.3) that
 $y^{(k)}(t) = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \Delta^k b_i B_i^{n-k}(t) \geq 0$ for $t \in [0, 1]$, $k = 0, 1, \dots, n$.

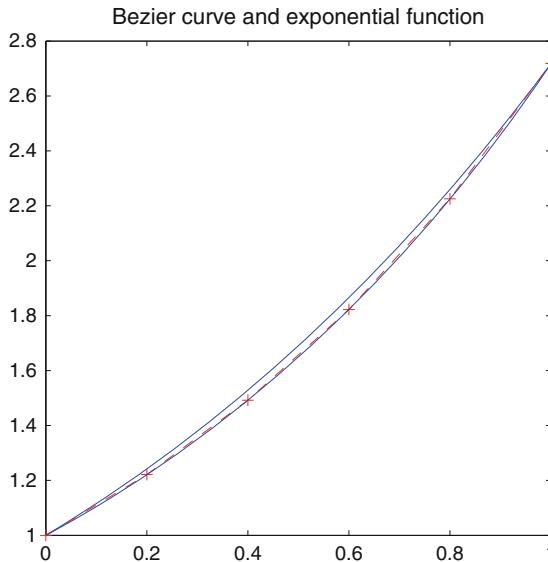


Fig. 7.6 Approximating e^x by a Bézier curve with 11 Bézier points

3. With $b_i = \exp(i/n)$ for $i = 0, 1, \dots, n$ we find

$$\Delta^k b_i = \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} \exp\left(\frac{i+j}{n}\right) = \exp\left(\frac{i}{n}\right) \left(\exp\left(\frac{1}{n}\right) - 1\right)^k > 0$$

for $i = 0, 1, \dots, n-k$, $k = 0, 1, \dots, n$. It follows that $y^{(k)}(t) > 0$ for $t \in [0, 1]$ by what we just showed

4.

```
clear
n=5;
x=0:1/n:1;
b=exp(x);
C=[x;b]';
XX=0:1/200:1;
titleb='Bezier curve and exponential function';
plotbezier(C,200,titleb)
hold on
plot(XX,exp(XX))
```

See (Fig. 7.6).

5. With $y(t) = 1 - 3t + 9t^3$ we find $y'(t) = -3 + 27t^2$ which vanishes for $t = 1/3$. Since $y''(t) = 54t > 0$ for $t > 0$, there is a local minimum at $t = 1/3$ that is a global minimum of $y(t)$ for $t \in [0, 1]$. But then $y(t) \geq y(1/3) = 1/3 > 0$. We find

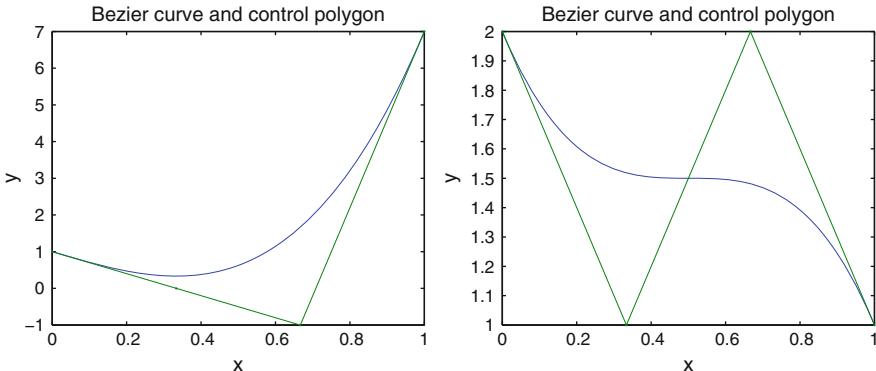


Fig. 7.7 Positive and monotone Bézier curves

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -3 \\ 0 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \\ 7 \end{bmatrix},$$

and b_2 is negative (Fig. 7.7 left).

6. With $y(t) = 2 - 3t + 6t^2 - 4t^3$ we find $y'(t) = -3 + 12t(1-t) \geq 0$ for $t \in [0, 1]$. We find

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 1 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -3 \\ 6 \\ -4 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}.$$

The b 's are not monotonically decreasing (Fig. 7.7 right).

7.5.4 Smooth Joining of Bézier Curves

Exercise 7.6

- \mathbf{P}_1 is of degree 3.
- Using (7.3), and writing $\mathbf{P}'_1(1) = \mathbf{P}'_2(1)$, we obtain $3\Delta\mathbf{a}_2 = 2\Delta\mathbf{b}_0$ which gives $\mathbf{b}_1 = \frac{3}{2}(\mathbf{a}_3 - \mathbf{a}_2) + \mathbf{b}_0$. Thus $\mathbf{b}_1 = (1, 5/2)$.
- We write $\mathbf{P}'_3(3) = \mathbf{P}'_1(0)$ and $\mathbf{P}''_3(3) = \mathbf{P}''_1(0)$ to obtain the two equations

$$3\Delta\mathbf{c}_2 = 3\Delta\mathbf{a}_0, \quad 3 \times 2\Delta^2\mathbf{c}_1 = 3 \times 2\Delta^2\mathbf{a}_0,$$

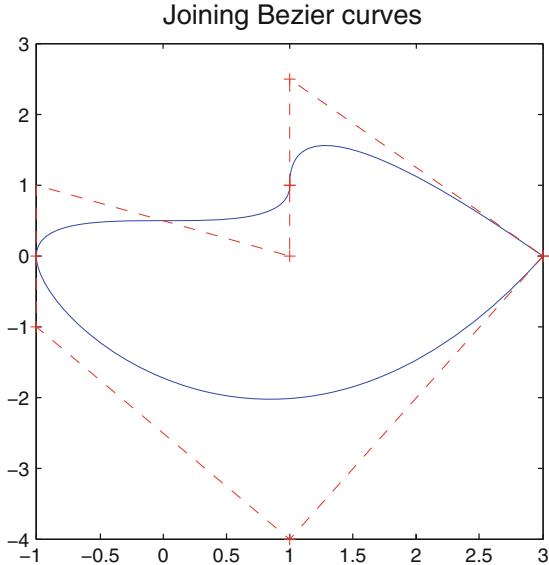


Fig. 7.8 Joining Bézier curves

which can be simplified into

$$\mathbf{c}_3 - \mathbf{c}_2 = \mathbf{a}_1 - \mathbf{a}_0, \quad \mathbf{c}_3 - 2\mathbf{c}_2 + \mathbf{c}_1 = \mathbf{a}_2 - 2\mathbf{a}_1 + \mathbf{a}_0.$$

Thus $\mathbf{c}_1 = (1, -4)$ and $\mathbf{c}_2 = (-1, -1)$.

4. The graph is given in Fig. 7.8.

Exercise 7.7

1. By the end point property for Bézier curves we have $\mathbf{P}_1(b) = \mathbf{d}_m$ and $\mathbf{P}_2(b) = \mathbf{e}_0$. Thus \mathbf{P} is continuous at zero if $\mathbf{d}_m = \mathbf{e}_0$.
2. By what we just showed \mathbf{P} is continuous at $t = b$. We find

$$\frac{d}{dx} \mathbf{P}_1(x) = \frac{d}{dx} \mathcal{B}(\mathbf{d}_0, \dots, \mathbf{d}_m) \left(\frac{x-a}{b-a} \right).$$

Therefore, by the tangent end point property (7.13) $\mathbf{P}'_1(b) = \frac{m}{b-a}(\mathbf{d}_m - \mathbf{d}_{m-1})$. Similarly $\mathbf{P}'_2(b) = \frac{n}{c-b}(\mathbf{e}_1 - \mathbf{e}_0)$. It follows that $\mathbf{P}'_1(b) = \mathbf{P}'_2(b)$ and the derivative of \mathbf{P} is continuous at $t = b$. The derivative $\mathbf{P}'_j(b)$ gives the direction of the tangent of \mathbf{P}_j at $t = b$ for $j = 1, 2$. Thus even if $\frac{m}{b-a} \neq \frac{n}{c-b}$ the tangents of \mathbf{P}_1 and \mathbf{P}_2 will point in the same direction, and therefore the tangent of \mathbf{P} is continuous at $t = b$.

3. By the differentiation formula (7.3) and the endpoint property for Bézier curves we find for $k = 0, 1, \dots, r$

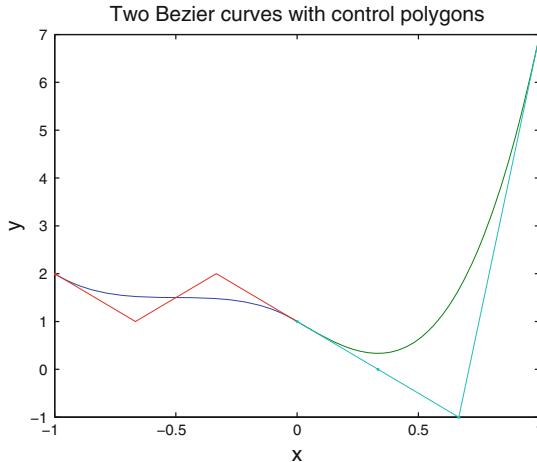


Fig. 7.9 A piecewise C^1 Bézier curve

$$\mathbf{P}_1^{(k)}(b) = \frac{n!}{(n-k)!} \sum_{i=0}^{n-k} \frac{\Delta^k \mathbf{d}_i}{(b-a)^k} B_i^{n-k}(1) = \frac{n!}{(n-k)!} \Delta^k \mathbf{d}_{n-k}/(b-a)^k.$$

Similarly, $\mathbf{P}_2^{(k)}(b) = \frac{n!}{(n-k)!} \Delta^k \mathbf{e}_0/(c-b)^k$. Since $b-a=c-b$ this shows that $\mathbf{P}_1^{(k)}(b) = \mathbf{P}_2^{(k)}(b)$.

4.

```
>> a1=[-1,1,0,0;2,-3,6,-4]';
>> a2=[0,1,0,0;1,-3,0,9]';
>> b1=powertobernstein(a1)
b1 =
    -1.0000    2.0000
   -0.6667    1.0000
   -0.3333    2.0000
      0    1.0000
>> b2=powertobernstein(a2)
b2 =
      0    1.0000
    0.3333         0
    0.6667   -1.0000
    1.0000    7.0000
>> B1=bezier2(b1,m);
>> B2=bezier2(b2,m);
>> plot(B1(:,1),B1(:,2),B2(:,1),B2(:,2),b1(:,1),...
        b1(:,2),b2(:,1),b2(:,2),'.-')
xlabel('x','Fontsize',14)
ylabel('y','Fontsize',14)
title('Two Bezier curves with control polygons',...
      'Fontsize',14)
```

See (Fig. 7.9).

5. It is C^0 since $\mathbf{d}_3 = (0, 1) = \mathbf{e}_0$ and C^1 since also $\mathbf{d}_3 - \mathbf{d}_2 = (\frac{1}{3}, -1) = \mathbf{e}_1 - \mathbf{e}_0$.

Chapter 8

Piecewise Polynomials, Interpolation and Applications

In this chapter, we continue the study of function approximation using the process of **polynomial interpolation** which is now done through a function consisting of pieces of polynomials glued together smoothly. In many cases, better results are obtained if such **piecewise polynomials** are used instead of a unique polynomial of high degree. These functions are used for approximation in many branches of science and engineering, see for example the introduction in Chap. 7.

In Exercises 8.1 and 8.2, we study an alternative to **cubic Hermite interpolation** known as **quadratic Hermite interpolation**. It consists of two quadratic pieces joined in a C^1 fashion, so that the combined piecewise quadratic function interpolates values and derivatives at the two end points. We recover the cubic Hermite interpolant in Exercise 8.3 with a piecewise cubic C^1 function interpolating given values and **first derivatives** (slopes) at a number of points. In Exercises 8.4 (first derivative boundary conditions) and 8.5 (not-a-knot boundary conditions), we show that it is possible to chose the slopes so that the interpolant, known as a **cubic spline interpolant**, has smoothness C^2 . The computed slopes in cubic spline interpolation are used in Exercises 8.6 and 8.7 as approximations to the first derivatives of the function at the data we are interpolating. The **second derivative** can also be approximated by the second derivative of the cubic spline, and this is explored in Exercise 8.8. In each case we estimate the convergence order numerically and obtain agreements with theoretical upper bound estimates. In Exercise 8.9, we use cubic splines to estimate the derivatives in order to approximate the **length of a curve** through numerical integration. Notice that numerical integration is treated more fully in the next chapter.

8.1 Quadratic and Cubic Hermite Interpolation

Given interpolating values y_1, y_2 and derivatives s_1, s_2 at two sites $a < b$, let $z \in (a, b)$ be a number called a **knot**. We seek a piecewise quadratic polynomial $g : [a, b] \rightarrow \mathbb{R}$ in the form

$$g(x) := \begin{cases} p_1(x), & \text{if } a \leq x < z, \\ p_2(x), & \text{if } z \leq x \leq b, \end{cases} \quad (8.1)$$

with $p_1, p_2 \in \mathbb{P}_2$ such that

$$g(a) = y_1, \quad g'(a) = s_1, \quad g(b) = y_2, \quad g'(b) = s_2. \quad (8.2)$$

Moreover, we require

$$p_1(z) = p_2(z), \quad p'_1(z) = p'_2(z). \quad (8.3)$$

which means that $g \in C^1[a, b]$. We will show that such a function g is uniquely given. It is called the **quadratic Hermite interpolant**. This is an alternative to the **cubic Hermite interpolant** considered in Exercise 6.2 which interpolates the same data (8.2), but consists of only one polynomial piece of degree 3.

Exercise 8.1. Quadratic and cubic Hermite interpolation.

- Verify that $g_2 : [0, 2] \rightarrow \mathbb{R}$ given by

$$g_2(x) := \begin{cases} 0, & \text{if } 0 \leq x < 1, \\ 16(x-1)^2, & \text{if } 1 \leq x \leq 2, \end{cases}$$

is a quadratic Hermite interpolant with a knot at $z = 1$, interpolating $f(x) = x^4$ at 0 and 2. Also find $g_3(x)$, the cubic Hermite interpolant interpolating the same data (see Review of Chap. 6 for the definition).

- Make a plot x4.m showing g_2 , g_3 , and f on the interval $[0, 2]$ (Fig. 8.1).

Review: We recall that a polynomial $p \in \mathbb{P}_d$ is said to be in **Bernstein form of degree d** on an interval $[\alpha, \beta]$, with $h := \beta - \alpha > 0$, if

$$p(x) = \sum_{j=0}^d c_j B_j^d \left(\frac{x-\alpha}{h} \right), \quad \text{where } B_j^d(t) = \binom{d}{j} t^j (1-t)^{d-j}, \quad d \geq 0. \quad (8.4)$$

We also recall that the coefficient c_j are uniquely given (see Chap. 7 and Exercise 11.3 for details, proofs and properties of the B_j^d). Moreover, for $d \geq 2$ (cf. Exercise 7.7) the values and derivatives at the endpoints are given by

$$p(\alpha) = c_0, \quad p(\beta) = c_d, \quad p'(\alpha) = \frac{d}{h}(c_1 - c_0), \quad p'(\beta) = \frac{d}{h}(c_d - c_{d-1}). \quad (8.5)$$

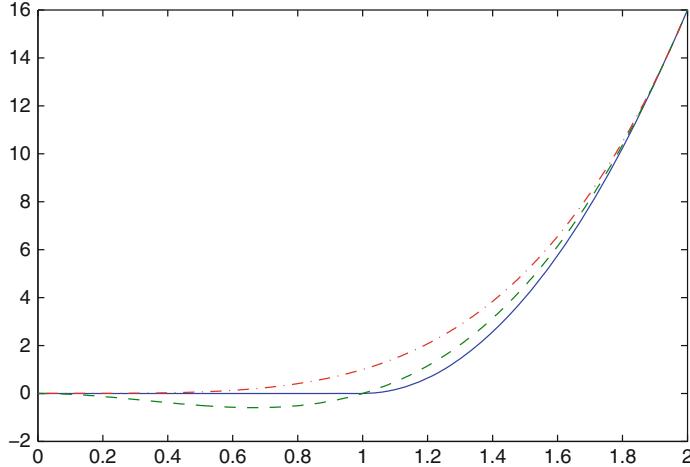


Fig. 8.1 Quadratic: —, and cubic Hermite interpolation: —— to $f(x) = x^4$: - · - on $[0, 2]$

Exercise 8.2. Quadratic and cubic Hermite interpolation using the Bernstein basis.

1. Consider cubic Hermite interpolation on $[a, b]$. Compute the coefficients (c_j) in the Bernstein form of degree 3 given by $p(x) = \sum_{j=0}^3 c_j B_j^3 \left(\frac{x-a}{b-a} \right)$ and satisfying (8.2). Then give the Bernstein form of g_3 in Exercise 8.1.
2. Show that in **quadratic** Hermite interpolation the coefficients in the **piecewise Bernstein form**

$$p_1(x) = \sum_{j=0}^2 c_{1j} B_j^2 \left(\frac{x-a}{z-a} \right), \quad p_2(x) = \sum_{j=0}^2 c_{2j} B_j^2 \left(\frac{x-z}{b-z} \right), \quad (8.6)$$

satisfying (8.2) and (8.3) are given by

$$\begin{aligned} c_{10} &= y_1, & c_{11} &= y_1 + \frac{h_1}{2} s_1, & c_{12} &= y, \\ c_{20} &= y, & c_{21} &= y_2 - \frac{h_2}{2} s_2, & c_{22} &= y_2. \end{aligned} \quad (8.7)$$

where $y := p_1(z) = p_2(z)$, (at this point unknown), $h_1 = z - a$, and $h_2 = b - z$.

3. Compute y ▷ *Math Hint*¹ ◁ and also give the particular case where $h_1 = h_2 = h$.
4. Give the Bernstein form of g_2 in Exercise 8.1.

¹ Use the C^1 condition at z .

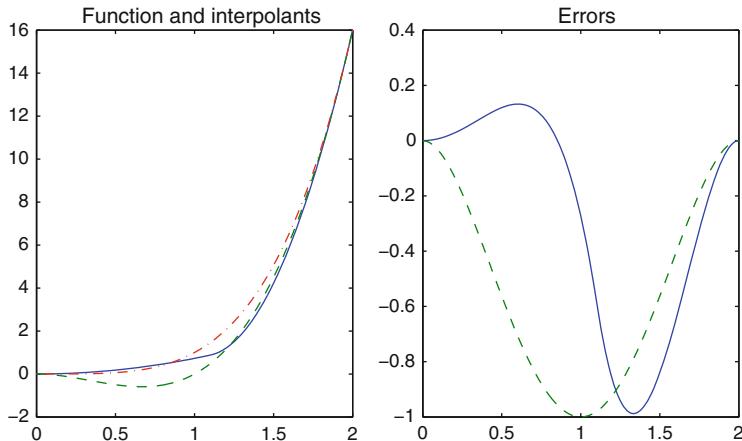


Fig. 8.2 On the left, Quadratic ($z=1.1$) and cubic Hermite interpolation to $f(x) = x^4$ on $[0, 2]$. On the right the errors $x^4 - g_2(x)$:— and $x^4 - g_3(x)$:— —

5. Write a MATLAB function $C=\text{QuadHermite}(a,b,z,y1,y2,s1,s2)$ which for given data given as above computes the vector of coefficients $C = [c_{1,0}, c_{1,1}, c_{1,2}, c_{2,1}, c_{2,2}]^T$ in (8.6). Tests:

```
>> QuadHermite(0,2,1,0,16,0,32)
ans =
    0         0         0         0        16
```

```
>> C=QuadHermite(0,2,1.1,0,16,0,32)
C =
    0         0       0.8800     1.6000    16.0000
```

6. Write a script `x4nonuniform.m` that produces two plots using the data in part 4. The first showing the quadratic Hermite interpolant g_2 computed in the Bernstein basis, the cubic Hermite interpolant g_3 and the function $f(x) = x^4$ on the interval $[0, 2]$ and the second the errors $f - g_2$ and $f - g_3$. Test with the previous data and $z = 1.1$ (Fig. 8.2)
7. Add a slider that gives the possibility of modifying z in $(0, 2)$ before recomputing g_2 and reploting the functions and errors. ▲ MATLAB Hint² ▷ Save as `x4nonuniformcursor.m`.

8.2 Cubic Spline Interpolation

The quadratic and cubic Hermite interpolants can be generalized to piecewise polynomials using an arbitrary number of pieces. We consider only the cubic case here. Given sites $\mathbf{x} := [x_1, \dots, x_{n+1}]$ with $a = x_1 < x_2 < \dots < x_{n+1} = b$, y-values

² Use the function `uicontrol`.

$\mathbf{y} := [y_1, \dots, y_{n+1}]$ and derivative values $\mathbf{s} := [s_1, \dots, s_{n+1}]$. In this problem we first (Exercise 8.3) construct a piecewise cubic Hermite interpolant g satisfying

$$g(x_i) = y_i, \quad g'(x_i) = s_i, \quad i = 1, 2, \dots, n + 1. \quad (8.8)$$

You will be asked to show that g has smoothness C^1 . Thus, for $r = 1$ the function g belongs to the space of piecewise cubic polynomials with smoothness C^r given by

$$\mathbb{P}_3^r := \{g \in C^r([a, b]) : g|_{[x_i, x_{i+1}]} \in \mathbb{P}_3, 0 \leq i \leq n\}, \quad r \geq 0. \quad (8.9)$$

next, in Exercise 8.4, we show that it is possible to choose the interior derivative values s_2, \dots, s_n so that $g \in C^2$. This function is known as a **cubic spline interpolant** (with first derivative boundary conditions), and it belongs to the space \mathbb{P}_3^2 of piecewise cubic polynomials with smoothness C^2 .

A function $g \in \mathbb{P}_3^2$ has the form

$$g(x) := \begin{cases} p_1(x), & \text{if } a \leq x < x_2, \\ p_2(x), & \text{if } x_2 \leq x < x_3, \\ \vdots \\ p_{n-1}(x), & \text{if } x_{n-1} \leq x < x_n, \\ p_n(x), & \text{if } x_n \leq x \leq b, \end{cases} \quad (8.10)$$

where each p_i is a polynomial of degree at most 3. We will represent each p_i in Bernstein form of degree 3 relative to the interval $[x_i, x_{i+1}]$. Thus with $B_0^3(t) = (1-t)^3, B_1^3(t) = 3t(1-t)^2, B_2^3(t) = 3t^2(1-t), B_3^3(t) = t^3$ we write

$$p_i(x) = \sum_{j=0}^3 c_{ij} B_j^3 \left(\frac{x - x_i}{h_i} \right), \quad \text{where } h_i = x_{i+1} - x_i. \quad (8.11)$$

Thirdly, in Exercise 8.5, we show how to avoid the knowledge of s_1 and s_{n+1} while keeping the C^2 -regularity and a good approximation order.

Exercise 8.3. Piecewise cubic Hermite interpolation using the Bernstein basis.

1. Show that for any increasing sites x and any data \mathbf{y} and \mathbf{s} there exist a unique piecewise cubic Hermite interpolant $g \in \mathbb{P}_3^1$. ▷ *Math Hint*³ ◁
2. Show that the coefficients c_{ij} of g are given by ▷ *Math Hint*⁴ ◁

$$c_{i,0} = y_i, \quad c_{i,1} = y_i + \frac{h_i}{3}s_i, \quad c_{i,2} = y_{i+1} - \frac{h_i}{3}s_{i+1}, \quad c_{i,3} = y_{i+1}. \quad (8.12)$$

³ Use Exercise 6.2.

⁴ Use Exercise 8.2 Part 1.

Review:

Consider now the C^2 case. The computation of the C^2 -spline is given in the following theorem (we will only show this in the case where $h := x_{i+1} - x_i$ is constant).

Theorem 8.1. *For all increasing sites $\mathbf{x} \in \mathbb{R}^{n+1}$, all choices of y -values $\mathbf{y} \in \mathbb{R}^{n+1}$ and end slopes s_1, s_{n+1} , there exists a unique cubic spline interpolant $g \in \mathbb{P}_3^2$ such that:*

$$\begin{aligned} g(x_i) &= y_i, \text{ for } 1 \leq i \leq n+1, \\ g'(x_1) &= s_1, \quad g'(x_{n+1}) = s_{n+1}. \end{aligned}$$

The function g is given by (8.10)–(8.12), where, in the case $h := x_{i+1} - x_i$ is constant, the unknown slopes are unique solutions of the linear tridiagonal system

$$\mathbf{A} \begin{bmatrix} s_2 \\ s_3 \\ \vdots \\ s_{n-1} \\ s_n \end{bmatrix} = \begin{bmatrix} 3(y_3 - y_1)/h - s_1 \\ 3(y_4 - y_2)/h \\ \vdots \\ 3(y_n - y_{n-2})/h \\ 3(y_{n+1} - y_{n-1})/h - s_{n+1} \end{bmatrix}, \quad (8.13)$$

$$\text{with } \mathbf{A} = \begin{bmatrix} 4 & 1 & 0 & & & \\ 1 & 4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & 4 & 1 & \\ & & 0 & 1 & 4 & \end{bmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}$$

♦

Exercise 8.4. Cubic spline interpolation with first derivative boundary conditions

1. Suppose $p(x) := \sum_{k=0}^3 c_k B_k^3(t)$, where $t = \frac{x-a}{h}$, and $h = b-a$. Compute $\frac{d^2 B_k^3}{dt^2}$ for $k = 0, 1, 2, 3$ then $\frac{d^2 p}{dx^2}$.
2. Show that the C^2 condition at x_i gives an equation connecting s_{i-1}, s_i, s_{i+1} and y_{i-1}, y_i, y_{i+1} . ▷ Math Hint ⁵ ◁ At this point the $s_i, i = 2, \dots, n$ are unknown...
3. Show that the vector $[s_2, \dots, s_n]^T$ must be a solution of the linear system (8.13).
4. We show the nonsingularity of the system (8.13) by proving that zero cannot be an eigenvalue. This is obtain by the following theorem

⁵ Compute $\frac{d^2 p_{i-1}(x_i)}{dx^2}$ and $\frac{d^2 p_i(x_i)}{dx^2}$.

Theorem 8.2 (Gershgorin's circle theorem).

If $m \in \mathbb{N}$, $\mathbf{A} \in \mathbb{C}^{m \times m}$ and λ is an eigenvalue of \mathbf{A} then

$$\lambda \in \bigcup_{i=1}^m \mathcal{D}_i, \text{ where } \mathcal{D}_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{j=1, j \neq i}^m |a_{ij}|\}. \quad (8.14)$$

Semyon Aranovich Gershgorin (1901–1933) was a belarus mathematician. He studied at the Petrograd Technological Institute, then became a Professor in 1930. He also taught at the Leningrad Mechanical Engineering Institute. His name is used for the Gershgorin circle theorem.



Prove Theorem 8.2 \triangleright *Math Hint*⁶ \triangleleft and deduce that the system (8.13) is nonsingular.

⊕ **Comment:** Proving that the system has a unique solution can also be done using that the matrix is strictly diagonally dominant (cf. Theorem 2.1). ⊕

⊕ **Comment:** One can show that if $f \in \mathcal{C}^4[a, b]$, $M_4 = \sup_{x \in [a, b]} |f^{(4)}(x)|$ and $h = \max_{i=1, \dots, n} |x_{i+1} - x_i|$, then

$$\forall x \in [a, b], \begin{cases} |f(x) - g(x)| & \leq \frac{5h^4}{384} M_4 \\ |f'(x) - g'(x)| & \leq \frac{h^3}{24} M_4 \\ |f''(x) - g''(x)| & \leq \frac{3h^2}{8} M_4 \end{cases} \quad (8.15)$$

**Exercise 8.5. Not-a-knot cubic spline interpolant**

Suppose s_1 et s_{n+1} are not known. One possibility could be to use, for examples, $g'(x_1) = g'(x_{n+1}) = 0$, or $g''(x_1) = g''(x_{n+1}) = 0$, but since in general one then loses the approximation order h^4 between the function and its interpolant near the end of the range, this is only satisfactory if the data is periodic. When imposing the conditions that g has smoothness C^3 at x_2 and x_n , we will prove numerically that the error will then continue to be of order 4 when sampling a regular function f . We begin by the proof of the following theorem for a uniform site.

Theorem 8.3. For $n \geq 3$, all increasing sites $\mathbf{x} \in \mathbb{R}^{n+1}$, all choices of y-values $\mathbf{y} \in \mathbb{R}^{n+1}$, there exists a unique cubic spline interpolant $g \in \mathbb{P}_3^2$ given by (8.10)–(8.12) such that:

$$\begin{aligned} g(x_i) &= y_i, \text{ for } 1 \leq i \leq n+1, \\ p_1'''(x_2) &= p_2'''(x_2), \quad p_{n-1}'''(x_n) = p_n'''(x_n). \end{aligned}$$

⁶ Show that if $\mathbf{Ax} = \lambda\mathbf{x}$ then $\lambda \in \mathcal{D}_i$, where $|x_i| = \max_j |x_j|$.

The unknown slopes \mathbf{s} are unique solutions of a linear system and in the case where $h := x_{i+1} - x_i$ is constant, the system is

$$\mathbf{A}\mathbf{s} = \frac{1}{h}\mathbf{B}\mathbf{y} \quad (8.16)$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 4 & 1 \\ \ddots & \ddots & \ddots \\ & 1 & 4 & 1 \\ & 1 & 0 & -1 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -2 & 4 & -2 \\ -3 & 0 & 3 \\ \ddots & \ddots & \ddots \\ & -3 & 0 & 3 \\ & -2 & 4 & -2 \end{bmatrix}, \quad \mathbf{s} = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_n \\ s_{n+1} \end{bmatrix},$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \\ y_{n+1} \end{bmatrix}.$$

1. Show (8.16). ▷ Math Hint⁷ ◁
2. Show that the system (8.16) can be written

$$\begin{cases} s_1 - s_3 = \frac{2}{h}(-y_1 + 2y_2 - y_3) \\ \bar{\mathbf{A}}\bar{\mathbf{s}} = \bar{\mathbf{b}}, \quad \bar{\mathbf{s}} := [s_2, \dots, s_n]^T \\ s_{n-1} - s_{n+1} = \frac{2}{h}(-y_{n-1} + 2y_n - y_{n+1}) \end{cases}, \quad (8.17)$$

where

$$\bar{\mathbf{A}} = \begin{bmatrix} 4 & 2 & 0 \\ 1 & 4 & 1 \\ \ddots & \ddots & \ddots \\ & 1 & 4 & 1 \\ & 0 & 2 & 4 \end{bmatrix} \in \mathbb{R}^{(n-1) \times (n-1)}.$$

Then show that the system (8.16) has a unique solution ▷ Math Hint⁸ ◁ (We do not need the explicit form of $\bar{\mathbf{b}}$ for this discussion). Prove Theorem 8.3.

3. MATLAB Programs: Given a function $f : [a, b] \rightarrow \mathbb{R}$, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n+1}$ with $x_i := a + (i-1)(b-a)/n$, $i = 1, 2, \dots, n+1$, $\mathbf{y} = f(\mathbf{x})$, and $\mathbf{X} \in \mathbb{R}^{m+1}$ with $X_j := a + (j-1)(b-a)/m$, $j = 1, 2, \dots, m+1$. The MATLAB function `Y=spline(x,y,X)` returns the values $Y = g(X)$ of the not-a-knot spline

⁷ Add two new equations and modify the first and last equations in (8.13).

⁸ Show that $\bar{\mathbf{A}}\bar{\mathbf{s}} = \bar{\mathbf{b}}$ has a unique solution first.

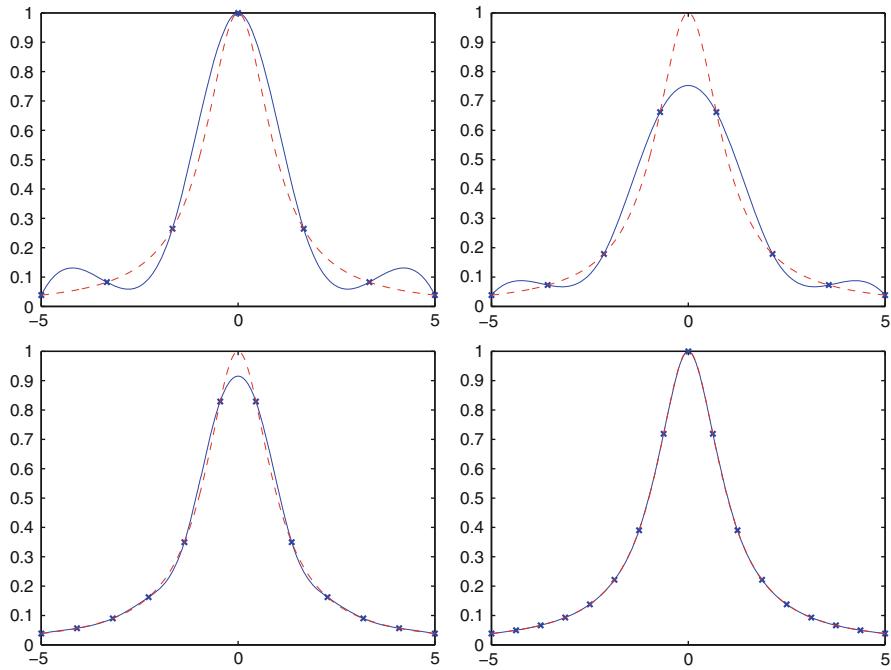


Fig. 8.3 Function f_2 and Cubic Spline interpolating at 6, 7, 11 and 16 equidistant points using not-a-knot boundary conditions

interpolant g satisfying $g(\mathbf{x}) = \mathbf{y} = f(\mathbf{x})$. In this exercise we are given the three functions

$$f_1(x) = e^x \text{ on } [0, 1], \quad f_2(x) = \frac{1}{1+x^2} \text{ on } [-5, 5],$$

$$f_3(x) = \begin{cases} x & \text{if } x < 0, \\ x/2 & \text{otherwise,} \end{cases} \text{ on } [-2, 2].$$

- a. Write a function $y=f1(x)$ that to given vector (or matrix) \mathbf{x} returns $f1(\mathbf{x})$. Same for $f2$ and $f3$.

```
>> f3([-1,1])
ans =
-0.5000    1.0000
```

- b. Write a function $\text{splineplot}(a, b, f, n, m)$ that calls spline and draws the graph of f and the spline. Try splineplot on the function f_2 for some values of n . Test $\text{splineplot}(-5, 5, @f2, 6, 1000)$ (Fig. 8.3);

©Comment: By increasing n , one can see that the Runge phenomenon does not appear (See Sect. 6.2.4). ☺

- c. Let $\mathbf{Y} = g(\mathbf{X})$ be a spline approximation to $f(\mathbf{X})$ and let $e := \|\mathbf{Y} - f(\mathbf{X})\|_\infty$. Write a function `e=splineerr(a,b,f,N,m)`, where $\mathbf{N} = [N_1, \dots, N_r]^T \in \mathbb{R}^r$ is a vector of r increasing positive integers. For each ℓ , one starts with $N_\ell + 1$ equidistant x -values, calculates $m + 1$ points $(\mathbf{X}, \mathbf{Y}_\ell = g(\mathbf{X}))$ on the corresponding not-a-knot spline interpolant and the corresponding error $e_\ell = \|\mathbf{Y}_\ell - f(\mathbf{X})\|_\infty$. Test on the functions f_1 and f_3 and $\mathbf{N} = [4, 8]$. ▷ MATLAB Hint⁹

```
>> N=[4 8], e=splineerr(0,1,@f1,N,1000)
N =
    4      8
e =
    1.0e-03 *
    0.2207    0.0165
```

```
>> N=[4 8], e=splineerr(-2,2,@f3,N,1000)
N =
    4      8
e =
    0.0481    0.0213
```

- d. Suppose the error behaves as $e_\ell \simeq c/N_\ell^q$ for some q . We can determine q as the slope of the straight line that is the least-squares fit to the data $(\log N_\ell, \log e_\ell)_{\ell=1}^r$. Extend the previous function into a function that plots $\log e$ as a function of $\log N$ and prints the approximation order q as the title of the plot. ▷ MATLAB Hint¹⁰ Test: `q=plotsplineorder(a,b,f,N,m)` on both f_1 and f_3 , with $N = [3, 5, 7, 10, 15, 20, 35, 50, 75, 100]^T$ (Fig. 8.4).

```
>> N=[3 5 7 10 15 20 35 50 75 100];
>> q=plotsplineorder(0,1,@f1,N,1000)
q =
    -3.9671
>> q=plotsplineorder(-2,2,@f3,N,1000)
q =
    -1.1380
```

©Comment: For the function f_1 the error behaves approximately as $O(h^4)$ since $f_1 \in C^\infty$, proving numerically that (8.15) is optimal. The function f_3 has a discontinuous first derivative at zero and the error behaves more like $O(h)$.



⁹ To compute $f(x)$ where `f.m` is a MATLAB function file, call `@f` in the input variables and `f(x)` in the program.

¹⁰ Use for example the MATLAB function `polyfit` to find q .

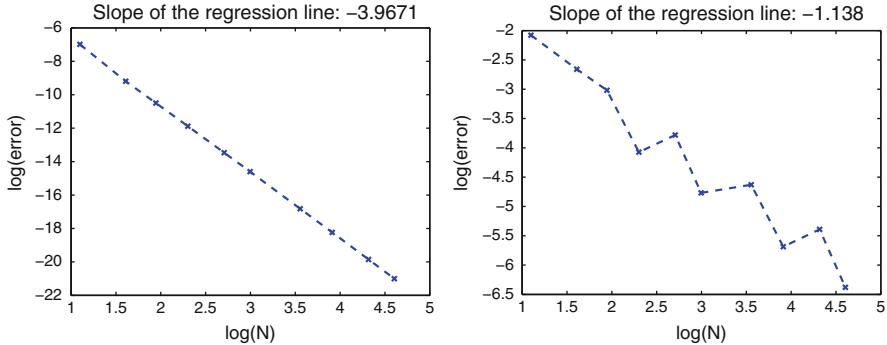


Fig. 8.4 Errors in spline approximations to functions f_1 (left) and f_3 (right)

8.3 Approximation of the Derivatives

Let $f(t)$ be a C^1 or C^2 function defined on $[a, b]$ with values in \mathbb{R} . We are looking for approximations to the first and second derivatives of f at $n + 1$ sampled points x_1, \dots, x_{n+1} in $[a, b]$.

For $n \in \mathbb{N}$, $n > 0$, we define $h := (b - a)/n$, $\mathbf{x} := [x_1, \dots, x_{n+1}]^T$ with $x_k = a + (k - 1)h$ for $k = 1, \dots, n + 1$ and $\mathbf{y} = f(\mathbf{x}) = [f(x_1), \dots, f(x_{n+1})]^T$. An approximation $\mathbf{s} \in \mathbb{R}^{n+1}$ of $f'(\mathbf{x}) := [f'(x_1), \dots, f'(x_{n+1})]^T$ is obtained when the system (8.16) is solved, i.e., $\mathbf{As} = \frac{1}{h}\mathbf{By}$. We can obtain approximations to the second derivative by solving the system

$$\mathbf{Ad} = \frac{1}{h}\mathbf{Bs}. \quad (8.18)$$

Exercise 8.6. Generate the A and B matrices

- Given $a, b, c \in \mathbb{R}$ and $n \in \mathbb{N}$ write a function $\text{A=tridiag}(a, b, c, n)$ that computes a tridiagonal matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with a on the sub diagonal, b on the main diagonal and c on the super diagonal. \triangleleft MATLAB Hint¹¹ \triangleright Test it on the

$$\text{matrix } \begin{bmatrix} 4 & -2 & 0 \\ 1 & 4 & -2 \\ 0 & 1 & 4 \end{bmatrix}.$$

```
>> A=tridiag(1,4,-2,3)
A =
    4     -2      0
    1      4     -2
    0      1      4
```

- Write a function $[\mathbf{A}, \mathbf{B}] = \text{splinematrices}(n)$ that generates the matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{(n+1) \times (n+1)}$ in (8.16)

¹¹ Use `diag` and `ones`.

```
>> [A,B]=splinematrices(3)
A =
    1      0     -1      0
    1      4      1      0
    0      1      4      1
    0      1      0     -1
B =
   -2      4     -2      0
   -3      0      3      0
    0     -3      0      3
    0     -2      4     -2
```

Exercise 8.7. New approximation of the first derivative A first construction was given in Exercise 6.11, using Lagrange Polynomials.

1. Write a function $[S, E] = \text{splineder}(a, b, f, df, n)$ that computes spline slopes $s \in \mathbb{R}^{n+1}$ and $e = \|s - df(x)\|_\infty$ by solving $As = \frac{1}{h}By$ using the construction of A and B in the previous exercise. Test using the functions $f_1(t) = t^3$ and $df_3(t) = 3t^2$, then $f_2(t) = \cosh t$ and $df_2(t) = \sinh t$.

```
>> [S,E]=splineder(0,2,@cosh,@sinh,4)
S =
    0.0214
    0.5161
    1.1727
    2.1419
    3.5745
E =
    0.0524
```

2. Write a program `derivativeorder` to study the error when modifying n . Start with an array $N = 20 : 10 : 200$. Compute a numerical estimation of the order of the method. Test with $a = 0$, $b = 2$ and the function $f_2(x) = \cosh(x)$. See Fig. 8.5.

©Comment: The approximation order is approximately h^3 in agreement with (8.15). ☺

Exercise 8.8. Approximation of the second derivative

1. Write a function $[D2, E] = \text{splineder2}(a, b, f, d2f, n)$ that computes approximations to the second derivatives $D2 \in \mathbb{R}^{n+1}$ of f using (8.18) and the error $e = \|D2 - d2f(x)\|_\infty$. Here $d2f$ is the exact second derivative of f . Test using the function $f(t) = \cosh t$.

```
>> [D2,E]=splineder2(0,2,@cosh,@cosh,4)
D2 =
    0.9282
    1.1010
    1.5755
```

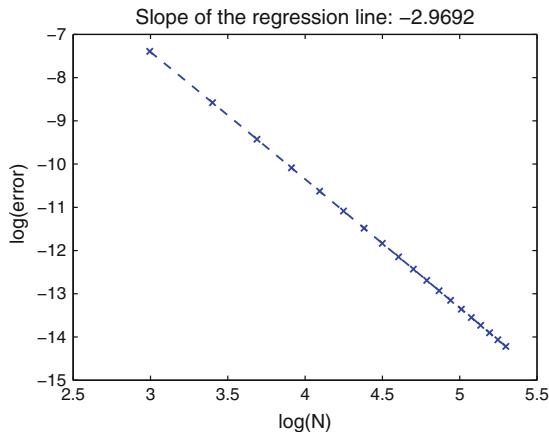


Fig. 8.5 Error in approximate first derivatives

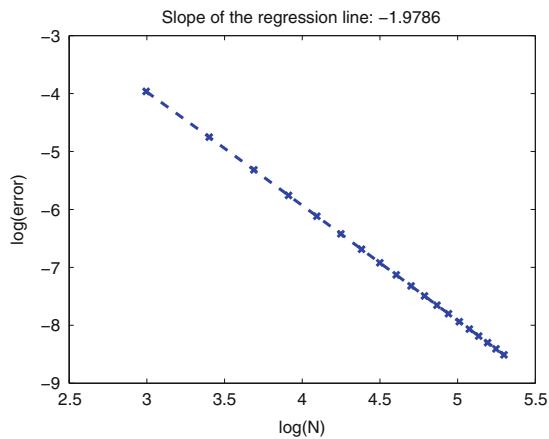


Fig. 8.6 Error in approximate second derivatives

$$\boxed{\begin{array}{l} 2.3516 \\ 3.4292 \\ E = \\ 0.3330 \end{array}}$$

2. Write a program to study the error when modifying n . Start with an array $N = 20 : 10 : 200$. Compute a numerical estimation of the order of the method. Test with $a = 0$, $b = 2$ and the function $f(x) = \cosh(x)$. See Fig. 8.6

©Comment: The approximation order is approximately h^2 in agreement with (8.15). ☺

8.4 Approximation of the Length of a Curve

We want to compute an approximation L of the length ℓ of the curve $y = f(x)$ with $x \in [a, b]$. If $f \in C^1[a, b]$, then $\ell = \int_a^b \sqrt{1 + f'(t)^2} dt$.

Beginning with a, b, n and the function f , we define again $h = (b - a)/n$, $x_k = a + (k - 1)h$, for $k = 1, \dots, n + 1$, $f(\mathbf{x}) = [f(x_1), \dots, f(x_{n+1})]^T$ and $f'(\mathbf{x}) = [f'(x_1), \dots, f'(x_{n+1})]^T$. An approximation $s \in \mathbb{R}^{n+1}$ of $f'(\mathbf{x})$ is computed by the cubic spline with not-a-knot boundary condition (see Theorem 8.3 for the system that gives s) from which we deduce the approximation $\mathbf{z} = [z_1, \dots, z_{n+1}]^T = \sqrt{1 + s^2}$ to $\sqrt{1 + f'(\mathbf{x})^2}$.

It remains to add a method to compute an approximation of the integral $I = \int_a^b \phi(t) dt$ where $\phi(t) = \sqrt{1 + f'(t)^2}$. We will consider two methods:

- The composite trapezoidal rule studied in Sect. 9.3:

$$L_T = \frac{h}{2} \left(z_1 + z_{n+1} + 2 \sum_{k=2}^n z_k \right). \quad (8.19)$$

- The composite Simpson rule for odd $n + 1$:

$$L_S = \frac{h}{3} \left(z_1 + z_{n+1} + 2 \sum_{k=1}^{\frac{n}{2}-1} z_{2k+1} + 4 \sum_{k=1}^{\frac{n}{2}} z_{2k} \right). \quad (8.20)$$

Exercise 8.9. Approximation of the length of a curve

1. Write functions `function T=trapsum(z, h)` and `function S=simpsum(z, h)` that computes the sums (8.19) and (8.20) without any loop.
↳ MATLAB Hint¹² ▷
Test with $z = (0, 1, 4, 9, 16)$.

```
>> trapsum(z,1)
ans =
    22
>> simpsum(z,1)
ans =
    21.3333
```

⌚Comment: Since the values are sampled from a quadratic function the value found by the composite Simpson rule is the exact value: $\int_0^4 t^2 dt = 64/3$. ☺

2. Write a function `[L, E] = Lapp(a, b, f, intrule, n, l)` that computes an approximation L to ℓ and the error $E := |L - \ell|$ where `intrule` has two

¹² A sum can be computed using `sum`.

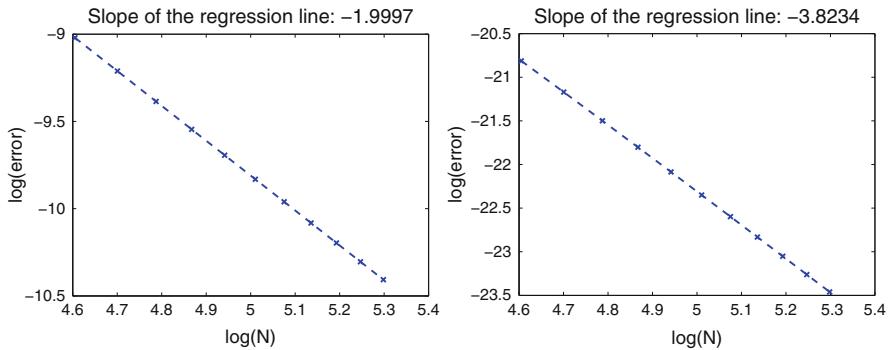


Fig. 8.7 Order of the length error with the composite trapezoidal (order 2) and Simpson (order 4) rules

parameters as in `trapsum` or `simpsum`. Use $a = 0$, $b = 2$, $f(x) = \cosh(x)$ and $n = 4$. ◁ MATLAB Hint¹³ ▷

```
>> [L,E]=Lapp(0,2,@cosh,@trapsum,4,sinh(2))
L =
    3.6931
E =
    0.0663
>> [L,E]=Lapp(0,2,@cosh,@simpsum,4,sinh(2))
L =
    3.6251
E =
    0.0017
```

3. You will now study of the error when modifying n . Compute a numerical approximation of the order of the methods using the two integration rules. Use $N = [100 : 10 : 200]$, $a = 0$, $b = 2$ and $f(x) = \cosh(x)$. Save as `orderlengtherror.m`. Results in Fig. 8.7.

8.5 Solutions

8.5.1 Quadratic and Cubic Hermite Interpolation

Solution of Exercise 8.1:

- Clearly g_2 is of the form (8.1) with $p_1(x) = 0$ and $p_2(x) = 16(x - 1)^2$ being quadratic polynomials. The interpolation conditions (8.2) hold since $g(0) = p_1(0) = 0$, $g'(0) = p'_1(0) = 0$, $g(2) = p_2(2) = 2^4 = f(2)$, and $g'(2) = p'_2(2) = 32 = f'(2)$. Moreover, $p_1(1) = 0 = p_2(1)$ and $p'_1(1) = 0 = p'_2(1)$ proving (8.3) as well.

¹³ The functions `splineder`, `tridiag` of Exercise 8.7 can be used.

$g_3(x)$ is a cubic polynomial such that $g_3(0) = g'_3(0) = 0$. Hence $g_3(x) = x^2(ux + v)$. From the two conditions $g_3(2) = 16$ and $g'_3(2) = 32$, we deduce u and v so that $g_3(x) = 4x^2(x - 1)$.

2. x4.m

```
clear
x=0:0.005:1;
xx=[x,x+1];
y2=[zeros(1,length(x)),16*x.^2];
y3=4*xx.*(xx-1);
y4=xx.^4;
plot(xx,y2,xx,y3,'-',xx,y4,'-')
```

Solution of Exercise 8.2:

1. We use (8.5) with $d = 3$. That $c_0 = y_1$ and $c_3 = y_2$ follows immediately. The formulas for c_1 and c_2 follow by solving $s_1 = p'(a) = \frac{3}{h}(c_1 - c_0)$ for c_1 and $s_2 = p'(b) = \frac{3}{h}(c_3 - c_2)$ for c_2 .

$$c_0 = y_1, \quad c_1 = y_1 + \frac{h}{3}s_1, \quad c_2 = y_2 - \frac{h}{3}s_2, \quad c_3 = y_3, \quad (8.21)$$

where $h = b - a$.

With $[y_1, y_2, s_1, s_2] = [0, 16, 0, 32]$ and $h = 2$ we then find $c_0 = c_1 = 0$, $c_3 = 16$, and $c_2 = 16 - \frac{2}{3} * 32 = -\frac{16}{3}$. Thus $g_3(x) = 16(B_3^3(\frac{x}{2}) - \frac{1}{3}B_2^3(\frac{x}{2}))$.

2. We now use (8.5) with $d = 2$. For p_1 , we find $c_{10} = y_1$, $c_{12} = y$, and c_{11} is obtained by solving $s_1 = \frac{2}{h_1}(c_{11} - c_{10})$. For p_2 , we obtain similarly $c_{20} = y$, $c_{22} = y_2$, and c_{21} follows by solving $s_2 = \frac{2}{h_2}(c_{22} - c_{21})$. Finally

$$\begin{aligned} c_{10} &= y_1, & c_{11} &= y_1 + \frac{h_1}{2}s_1, & c_{12} &= y, \\ c_{20} &= y, & c_{21} &= y_2 - \frac{h_2}{2}s_2, & c_{22} &= y_2. \end{aligned} \quad (8.22)$$

3. To obtain y , we solve the equation $p'_1(z) = p'_2(z)$. Then using (8.5) on $[0, z]$ then $[z, 2]$, we deduce that $\frac{2}{h_1}(c_{12} - c_{11}) = \frac{2}{h_2}(c_{21} - c_{20})$ that can be also written $\frac{2}{h_1}(y - y_1 - \frac{h_1}{2}s_1) = \frac{2}{h_2}(y_2 - \frac{h_2}{2}s_2 - y)$. Solving for y we find $(h_1 + h_2)y = h_1(y_2 - \frac{h_2}{2}s_2) + h_2(y_1 + \frac{h_1}{2}s_1)$ or

$$y = \frac{h_1y_2 + h_2y_1 - \frac{1}{2}h_1h_2(s_2 - s_1)}{h_1 + h_2}, \quad (8.23)$$

If $h_1 = h_2 = h$, then

$$y = \frac{1}{2}(y_2 + y_1) - \frac{h}{4}(s_2 - s_1), \quad h_1 = h_2 = h. \quad (8.24)$$

4. With $[y_1, y_2, s_1, s_2] = [0, 16, 0, 32]$ and $h_1 = h_2 = 1$, we find $y = \frac{1}{2}(16 + 0) - \frac{1}{4}(32 - 0) = 0$ from (8.24). But then $c_{1j} = 0$ for $j = 0, 1, 2$. For p_2 , we find

$c_{20} = y = 0$, $c_{21} = 16 - \frac{1}{2} \times 32 = 0$, and $c_{22} = 16$. Thus $p_1(x) = 0$ and $p_2(x) = 16B_2^2(x-1)$.

5. QuadHermite.m

```
function C=QuadHermite(a,b,z,y1,y2,s1,s2)
h1=z-a; h2=b-z;
C=[y1,y1+h1*s1/2,...(h1*y2+h2*y1-h1*h2*(s2-s1)/2)/(h1+h2),y2-h2*s2/2,y2]';
```

6. x4nonuniform.m

```
clear
a=0;b=2;
y1=0;y2=16;s1=0;s2=32;
n=500;
z=1.1;
C=QuadHermite(a,b,z,y1,y2,s1,s2);
t=0:1/n:1;
B20=(1-t).^2;B21=2*t.* (1-t);B22=t.^2;
x2=[a:(z-a)/n:z,z:(b-z)/n:b];
g2=[C(1)*B20+C(2)*B21+C(3)*B22,...C(3)*B20+C(4)*B21+C(5)*B22];
B30=(1-t).^3;B31=3*t.* (1-t).^2;
B32=3*t.^2.* (1-t);B33=t.^3;
x3=2*t;
g3=y1*B30+(y1+(b-a)*s1/3)*B31...
+(y2-(b-a)*s2/3)*B32+y2*B33;
f2=x2.^4;f3=x3.^4;

subplot(1,2,1)
plot(x2,g2,x3,g3,'—',x3,f3,'-')
title('Function and interpolants')
subplot(1,2,2)
plot(x2,g2-f2,x3,g3-f3,'—')
title('Errors')
```

7. x4nonuniformcursor.m

```
clear
a=0;b=2;
y1=0;y2=16;s1=0;s2=32;
n=500;
z=1.1;
valz=uicontrol('Style','slider','Min',-0.1,'Max',...
    1.99,'Position',[150 5 300 20],'value',z);
while z>0
    C=QuadHermite(a,b,z,y1,y2,s1,s2);
    t=0:1/n:1;
    B20=(1-t).^2;B21=2*t.* (1-t);B22=t.^2;
    x2=[a:(z-a)/n:z,z:(b-z)/n:b];
    g2=[C(1)*B20+C(2)*B21+C(3)*B22,...C(3)*B20+C(4)*B21+C(5)*B22];
```

```

B30=(1-t).^3; B31=3*t.* (1-t).^2;
B32=3*t.^2.* (1-t); B33=t.^3;
x3=2*t;
g3=y1*B30+(y1+(b-a)*s1/3)*B31...
+(y2-(b-a)*s2/3)*B32+y2*B33;

f2=x2.^4; f3=x3.^4;

subplot(1,2,1)
plot(x2,g2,x3,g3,'—',x3,f3,'-.')
title([ 'z = ',num2str(z)])
subplot(1,2,2)
plot(x2,g2-f2,x3,g3-f3,'—')
title('Errors')

pause(1)
z=get(valz,'value');
end

```

8.5.2 Cubic Spline Interpolation

Solution of Exercise 8.3:

- For $i = 1, \dots, n$, it follows from Exercise 6.2 that there exists a unique cubic polynomial p_i such that $p_i(x_i) = y_i$, $p'_i(x_i) = s_i$, $p_i(x_{i+1}) = y_{i+1}$ and $p'_i(x_{i+1}) = s_{i+1}$. Thus we can construct g in the form (8.10). Since g is C^∞ on (x_i, x_{i+1}) , to prove that $g \in C^1$, it is enough to show that we have continuity of value and derivative at each x_i . But this follows immediately since $p_{i-1}(x_i) = p_i(x_i) = y_i$ and $p'_{i-1}(x_i) = p'_i(x_i) = s_i$, $i = 2, \dots, n$. Moreover, g is unique since each p_i is unique.
- This follows immediately from Exercise 8.2 Part 1.

Solution of Exercise 8.4:

- We find $\frac{d^2 B_0^3}{dt^2} = 6(1-t)$, $\frac{d^2 B_1^3}{dt^2} = 6(3t-2)$, $\frac{d^2 B_2^3}{dt^2} = 6(1-3t)$, $\frac{d^2 B_3^3}{dt^2} = 6t$. Moreover, with $t = \frac{x-a}{h}$,

$$\frac{d^2 p}{dx^2} = \frac{1}{h^2} \frac{d^2 p}{dt^2} = \frac{6}{h^2} (c_0(1-t) + c_1(3t-2) + c_2(1-3t) + c_3t).$$

- Using the values computed by (8.12):

$$c_{i,0} = y_i, \quad c_{i,1} = y_i + \frac{h}{3}s_i, \quad c_{i,2} = y_{i+1} - \frac{h}{3}s_{i+1}, \quad c_{i,3} = y_{i+1}$$

we find

$$\begin{aligned}\frac{d^2 p_i}{dx^2}(x_i) &= \frac{6}{h^2}(c_{i,0} - 2c_{i,1} + c_{i,2}) = \frac{6}{h} \left(\frac{y_{i+1} - y_i}{h} - \frac{2}{3}s_i - \frac{1}{3}s_{i+1} \right), \\ \frac{d^2 p_i}{dx^2}(x_{i+1}) &= \frac{6}{h^2}(c_{i,1} - 2c_{i,2} + c_{i,3}) = \frac{6}{h} \left(-\frac{y_{i+1} - y_i}{h} + \frac{1}{3}s_i + \frac{2}{3}s_{i+1} \right).\end{aligned}$$

The C^2 condition $\frac{d^2 p_{i-1}}{dx^2}(x_i) = \frac{d^2 p_i}{dx^2}(x_i)$ then leads to the equation

$$3\frac{y_{i+1} - y_i}{h} - 2s_i - s_{i+1} = -3\frac{y_i - y_{i-1}}{h} + s_{i-1} + 2s_i. \quad (8.25)$$

3. Equation (8.25) can be rearranged as follows

$$s_{i-1} + 4s_i + s_{i+1} = 3\frac{y_{i+1} - y_{i-1}}{h}, \quad i = 2, \dots, n.$$

We notice that in the first equation for $i = 2$, the term s_1 is known and can be moved to the right part of the equality. Similarly with the last equation and s_{n+1} . This implies (8.13) and shows the existence part of Theorem 8.1.

4. Let us prove the Gershgorin's circle theorem. For the eigenvalue $\lambda \in \mathbb{C}$ and corresponding eigenvector $\mathbf{x} \in \mathbb{C}^m$, let i be such that $|x_i| = \max_j |x_j|$. Then $x_i \neq 0$ since $\mathbf{x} \neq \mathbf{0}$ and we have $\sum_j a_{ij}x_j = \lambda x_i$ or $(\lambda - a_{ii})x_i = \sum_{j \neq i} a_{ij}x_j$. Dividing by x_i and taking absolute values we find

$$|\lambda - a_{ii}| = \left| \sum_{j \neq i} a_{ij}x_j/x_i \right| \leq \sum_{j \neq i} |a_{ij}| |x_j/x_i| \leq \sum_{j \neq i} |a_{ij}|.$$

Thus $\lambda \in \mathcal{D}_i$ and the theorem is proved.

For the system (8.13) we find $\mathcal{D}_i = \{z \in \mathbb{C} : |z - 4| \leq 2\}$ except for the first and last row where $\mathcal{D}_i = \{z \in \mathbb{C} : |z - 4| \leq 1\}$. We conclude that zero is not an eigenvalue since $0 \notin \mathcal{D}_j$ for any j . It follows that the matrix A is nonsingular and the system has a unique solution.

Solution of Exercise 8.5:

1. The equations 2 to n of (8.16) are deduced from the C^2 -condition at x_i (see Exercise 8.4 or system (8.13)).

For the first and last equations, we begin by the computation of the third derivatives of the Bernstein basis polynomials which are given by

$$\frac{d^3}{dt^3} B_0^3(t) = -6, \quad \frac{d^3}{dt^3} B_1^3(t) = 18, \quad \frac{d^3}{dt^3} B_2^3(t) = -18, \quad \frac{d^3}{dt^3} B_3^3(t) = 6,$$

Using (8.10)–(8.12), we obtain $\frac{d^3 p_i}{dx^3}(x) = \frac{6}{h^3}(-c_{i,0} + 3c_{i,1} - 3c_{i,2} + c_{i,3})$ where

$$c_{i,0} = y_j, \quad c_{i,1} = y_j + \frac{h}{3}s_j, \quad c_{i,2} = y_{j+1} - \frac{h}{3}s_{j+1}, \quad c_{i,3} = y_{j+1}, \quad i = 1, \dots, n,$$

For $i = 2$ and $i = n$ the conditions $p_i'''(x_i) = p_{i-1}'''(x_i)$ imply

$$\begin{aligned} & -y_i + 3\left(y_i + \frac{h}{3}s_i\right) - 3\left(y_{i+1} - \frac{h}{3}s_{i+1}\right) + y_{i+1} \\ &= -y_{i-1} + 3\left(y_{i-1} + \frac{h}{3}s_{i-1}\right) - 3\left(y_i - \frac{h}{3}s_i\right) + y_i, \end{aligned}$$

which simplifies to $-2y_{i-1} + 4y_i - 2y_{i+1} = h(s_{i-1} - s_{i+1})$ giving the first and last equation in (8.17).

2. To obtain the first equation of the system $\bar{\mathbf{A}}\bar{\mathbf{s}} = \bar{\mathbf{b}}$, we simply subtract the first equation from the second one in (8.16). Similarly we obtain the last equation in $\bar{\mathbf{A}}\bar{\mathbf{s}} = \bar{\mathbf{b}}$ by adding the last two equations in (8.16). The other equations in (8.17) are the equations in (8.16). Now, the system $\bar{\mathbf{A}}\bar{\mathbf{s}} = \bar{\mathbf{b}}$ has a unique solution since $\bar{\mathbf{A}}$ is strictly diagonally dominant (or using the Gershgorin's circle theorem). In particular, this gives the solutions s_3 (respect. s_{n-1}) from which we deduce s_1 (respect. s_{n+1}) using the first equation (respect. last) of (8.17). Since (8.17) is obtained from (8.16) by row operations that preserves nonsingularity and the system (8.16) is nonsingular if and only if (8.17) is nonsingular. The proof of Theorem 8.3 follows exactly as the proof of Theorem 8.1.

3. a. f1.m, f2.m, f3.m

```
function y=f1(x)
y=exp(x);
function y=f2(x)
y=1./(1+x.*x);
function y=f3(x)
y=x; i=find(x>0); y(i)=y(i)/2;
```

- b. splineplot.m

```
function splineplot(a,b,f,n,m)
x=a:(b-a)/n:b;
y=f(x);
X=a:(b-a)/m:b;
Yf=f(X);
Ys=spline(x,y,X);
plot(x,y,'.b',X,Ys,'b',X,Yf,'r—')
```

- c. splineerr.m

```
function e=splineerr(a,b,f,N,m)
for l=1:length(N)
    n=N(l);
    x=a:(b-a)/n:b;
    y=f(x);
    X=a:(b-a)/m:b;
    Yf=f(X);
    Ys=spline(x,y,X);
    e(l)=norm(Ys-Yf,inf);
end
```

d. plotapporder.m

```
function q=plotsplineorder(a,b,f,N,m)
for l=1:length(N)
    n=N(l);
    x=a:(b-a)/n:b;
    y=f(x);
    X=a:(b-a)/m:b;
    Yf=f(X);
    Ys=spline(x,y,X);
    e(l)=norm(Ys-Yf,inf);
end
plot(log(N),log(e),'x—')
c=polyfit(log(N),log(e),1);
q=c(1);
title([ 'Slope of the regression line: ',num2str(q)]);
xlabel('log(N)');
ylabel('log(error)');
```

8.5.3 Approximation of the Derivatives**Solution of Exercise 8.6:**

1. tridiag.m

```
function A=tridiag(a,b,c,n)
A=a*diag(ones(n-1,1),-1)+b*diag(ones(n,1))...
+c*diag(ones(n-1,1),1);
```

2. splinematrices.m

```
function [A,B]=spline_matrices(n)
A=tridiag(1,4,1,n+1);
A(1,1:3)=[1 0 -1];A(n+1,n-1:n+1)=[1 0 -1];
B=tridiag(-3,0,3,n+1);
B(1,1:3)=[-2 4 -2];B(n+1,n-1:n+1)=[-2 4 -2];
```

Solution of Exercise 8.7:

1. splineder.m

```
function [S,E]=splineder(a,b,f,df,n)
h=(b-a)/n;
x=(a:h:b)';
[A,B]=spline_matrices(n);
S=A\B*f(x)/h;
E=norm(S-df(x),inf);
```

2. derivativeorder.m

```
N=20:10:200;
E=zeros (length (N) ,1);
for i=1:length (N)
    [S,E(i)]=splineder (0 ,2 ,@cosh ,@sinh ,N(i));
end
q=plotapporder (N,E);
```

Solution of Exercise 8.8:

1. splineder2.m

```
function [D2,E]=splineder2 (a ,b ,f ,d2f ,n)
h=(b-a)/n;
x=(a:h:b)';
[A,B]=splinematrices (n);
S=A\ (B*f (x))/h;
D2=A\ (B*S)/h;
E=norm (D2-d2f (x) ,inf );
```

2.

```
clear
N=20:10:200;
for i=1:length (N)
    [S,E(i)]=splineder2 (0 ,2 ,@cosh ,@cosh ,N(i));
end
plot (log (N) ,log (E) , 'x—')
c=polyfit (log (N) ,log (E) ,1);
q=c (1);
title ([ 'Slope of the regression line: ' ,num2str (q)]);
xlabel ('log (N)');
ylabel ('log (error)' );
```

8.5.4 Approximation of the Length of a Curve**Solution of Exercise 8.9:**

1. trapsum.m

```
function T=trapsum (z ,h)
n=length (z)-1;
T=h/2*(z(1)+z(n+1)+2*sum (z (2:n )));
```

simpsum.m

```
function S=simpsum (z ,h)
n=length (z)-1;
S=h/3*(z(1)+z(n+1)+2*sum (z (3:2:n-1))+4*sum (z (2:2:n )));
```

2. Lapp.m

```
function [L,E]=Lapp(a,b,f,inrule,n,l)
h=(b-a)/n;
x=(a:h:b)';
A=diag(ones(n,1),-1)+4*diag(ones(n+1,1))...
+diag(ones(n,1),1);
A(1,1:3)=[1 0 -1];A(n+1,n-1:n+1)=[1 0 -1];
B=-3*diag(ones(n,1),-1)+3*diag(ones(n,1),1);
B(1,1:3)=[-2 4 -2];B(n+1,n-1:n+1)=[-2 4 -2];
S=A\ (B*f(x))/h;
Z=sqrt(1+S.^2);
L=inrule(Z,h);
E=abs(L-l);
```

3. orderlengtherror.m

```
N=[100:10:200];
inrule=@simpsum;
for i=1:length(N)
    [~,E(i)]=Lapp(0,2,@cosh,inrule,N(i),sinh(2));
end
plot(log(N),log(E),'x—')
c=polyfit(log(N),log(E),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(N)');
ylabel('log(error)');
```

Chapter 9

Approximation of Integrals

The usual MATLAB commands for the computation of integrals are `quad` and `quadl`. Let us test `quad` on $\int_0^1 f_0(x)dx = 5/18$, where $f_0(x) := |x - 1/3|$. We obtain

```
>> quad('f0',0,1)-5/18
ans =
-3.2746e-07
```

With `quadl`, we find

```
>> quadl('f0',0,1)-5/18
ans =
-8.9150e-08
```

These errors are far from the `eps`, the floating-point relative accuracy in MATLAB. With the call `quad(fun, a, b, tol)`, one can modify the tolerance and use an absolute error tolerance `tol` instead of the default which is `1.0e-6`. With `tol = 10-15` we obtain

```
>> quad('f0',0,1,1e-15)-5/18
ans =
-3.2196e-15
```

In this chapter, we first test different elementary quadrature formulas of the form $\sum_{i=1}^n w_i f(x_i)$ for the approximation of $\int_a^b f(t) dt$. We say that a quadrature rule has (exact) **degree of precision** d if it is exact for all polynomials of degree $\leq d$ and not exact for some polynomial of degree $d + 1$.

In Exercise 9.1, instead of computing the integral of the function, we compute the integral of its Lagrange interpolant p_n of degree $n - 1$ (See Chap. 6). This is tested for small values of n with the use of MATLAB to find the degree of precision in Exercise 9.2. In Exercise 9.4, we show how to improve the degree of precision by a “good” choice of the sampled points, while Exercise 9.5 is a detour to compute a double integral.

Then, we are concerned with the approximation of $\int_a^b w(t)f(t) dt$ where $w(t)$ is a fixed **weight function**. After a brief review, inner products are used in Exercises 9.6 and 9.7 to obtain accurate results. In Exercise 9.8, we compare two approximations.

A section is devoted to the Monte-Carlo method that uses random numbers to approximate multiple integrals. In Exercise 9.9, we compute an approximation of π by counting random points in a disk and in Exercise 9.10, we compute volumes of ellipsoids, and also intersections and unions of ellipsoids.

We go back to the trapezoidal rule which was introduced at the beginning of the chapter. We first study the basic rule in Exercise 9.11, and then the composite rule in Exercise 9.12. A MATLAB program is given in Exercise 9.13. Finally an approximation to the solution of an integral equation is computed in Exercise 9.14.

The Sect. 9.4 begins with Exercise 9.15 giving the proof of the Euler-Maclaurin formula. This formula provides the bases for the Romberg method (Exercise 9.16) which gives improved approximations starting with the composite trapezoidal rule.

We present a global quadrature rule in Exercise 9.17 that is used to solve an integral equation in Exercise 9.18.

Review: The following notation and terminology are used:

1. \mathbb{P}_n is the space of polynomials of degree at most n ,
2. **nodes or sites:** $x = [x_1, \dots, x_n]$ where $x_1 < x_2 < \dots < x_n$
3. **node polynomial** $\Pi(x) := \prod_{i=1}^n (x - x_i)$
4. $I(f) := \int_a^b f(x) dx$ or $I(f) := \int_a^b w(x)f(x) dx$. These are the integrals to be approximated. The fixed **weight function** w is integrable and nonnegative on $[a, b]$.
5. $J(f) := \sum_{i=1}^n w_i f(x_i)$, n point **quadrature rule**. The w_i are called **weights**.
6. A quadrature rule has **degree of precision** d if $I(p) = J(p)$ for any $p \in \mathbb{P}_d$. The rule has **exact degree of precision** d if it has degree of precision d and $I(p) \neq J(p)$ for at least one $p \in \mathbb{P}_{d+1}$.
7. The **truncation error** is defined by $R(f) := I(f) - J(f)$.
8. Recall that the **Lagrange basis** of degree $n - 1$ is given by (cf. Chap. 6)

$$\ell_i(x) := \prod_{\substack{j=1 \\ j \neq i}}^n \frac{t - x_j}{x_i - x_j}, \quad i = 1, \dots, n.$$

9. If $f \in C^n[a, b]$ and $p_n \in \mathbb{P}_{n-1}$ satisfies $p_n(x_i) = f(x_i)$, $i = 1, \dots, n$ then there is a point ξ_x in the smallest interval containing $[a, b]$ and every x_j such that (cf. Chap. 6)

$$\int_a^b f(x) dx = \int_a^b p_n(x) dx + \frac{1}{n!} \int_a^b f^{(n)}(\xi_x) \Pi(x) dx. \quad (9.1)$$

If f_i is a shorthand for $f(x_i)$, with $x_i = a + (i - 1)\frac{b-a}{n-1}$, we obtain the first Newton-Cotes formulae

n	Name	Formula	Error term
2	Mid-point rule	$(b-a)f_{3/2}$	$\frac{(b-a)^3}{24} f^{(2)}(\xi)$
2	Trapezoid rule	$\frac{b-a}{2}(f_1 + f_2)$	$-\frac{(b-a)^3}{12} f^{(2)}(\xi)$
3	Simpson's rule	$\frac{b-a}{6}(f_1 + 4f_2 + f_3)$	$-\frac{(b-a)^5}{2880} f^{(4)}(\xi)$

Roger Cotes (1682–1716) was an English mathematician, known for working closely with Isaac Newton by proofreading the second edition of his famous book, the Principia. He also invented the quadrature formulas known as Newton-Cotes formulas and first introduced what is known today as Euler's formula.



Thomas Simpson (1710–1761) was a British mathematician, inventor and eponym of Simpson's rule to approximate definite integrals. The attribution, as often in mathematics, can be debated: this rule had been found 100 years earlier by Johannes Kepler. Apparently, the method that is known as Simpson's rule was well known and used earlier by Bonaventura Cavalieri.



The weighted rules are presented in Sect. 9.1.2. ▲

9.1 Basic Quadrature Rules

9.1.1 Quadrature Rules and Degree of Precision

Exercise 9.1. Lagrange polynomial for integration.

Suppose a function $f : [a, b] \rightarrow \mathbb{R}$ is only known at certain distinct sites $x = [x_1, \dots, x_n]$, where $x_i \in [a, b]$, for $i = 1, \dots, n$. Let $p_n \in \mathbb{P}_{n-1}$ be the Lagrange interpolating polynomial at these sites.

1. Show that the basic quadrature $J(f) := \int_a^b p_n(t) dt$ satisfies

$$J(f) = \sum_{j=1}^n w_j f(x_j) \text{ where the } w_j \text{s depend on the Lagrange basis.} \triangleright \text{Math}$$

Hint¹ ◁

2. Show that $J(f)$ has degree of precision at least $n - 1$. ▷ *Math Hint²* ◁
3. Show that if $f \in \mathcal{C}^n([a, b])$, then the truncation error can be bounded in terms of the nodal polynomial as follows:

¹ See Exercise 6.3.

² Express $q \in \mathbb{P}_{n-1}$ in terms of the Lagrange basis $\{\ell_1, \dots, \ell_n\}$ to show that $I(q) = J(q)$.

$$|R(f)| \leq \frac{1}{n!} \max_{t \in [a,b]} |f^{(n)}(t)| \int_a^b |\Pi_n(t)| dt. \quad (9.2)$$

▷ *Math Hint*³ ◁

Exercise 9.2. Basic quadratures for small n .

We use the notations of the previous exercise with $[a, b] = [-1, 1]$. For each rule find $J(f)$ and use (9.2) to bound the error $R(f) = I(f) - J(f)$.

1. **Mid-point rule** $J_{MP}(f) : n = 1$ and $x_1 = 0$.
2. **Trapezoidal rule** $J_T(f) : n = 2$, $x_1 = -1$ and $x_2 = 1$.
3. **Simpson's rule** $J_S(f) : n = 3$, $x_1 = -1$, $x_2 = 0$ and $x_3 = 1$.
4. Show that $J_S(f) = \frac{2J_{MP}(f) + J_T(f)}{3}$.

⊕ **Comment:** The corresponding formulas for an arbitrary interval $[a, b]$ are given in the review, with a formula for the truncation error. ⊕

Exercise 9.3. Two basic quadrature rules. In the following, for a function $f : [a, b] \rightarrow \mathbb{R}$, f_i is a shorthand for $f(x_i)$, with $x_i = a + (i-1)(b-a)/(n-1)$.

1. For $n = 4$, consider **Simpson's 3/8 rule**

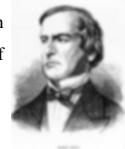
$$J_{S38}(f) = \frac{b-a}{8} (f_1 + 3f_2 + 3f_3 + f_4).$$

Choose the interval $[0, 1]$. Find the exact degree of precision. The error is given by $R_{S38}(f) = c_{S38} f^{(4)}(\xi)$. Find c_{S38} using MATLAB and a polynomial for f .

2. Same questions with **Boole's rule** for $n = 5$.

$$J_B(f) = \frac{b-a}{90} (7f_1 + 32f_2 + 12f_3 + 32f_4 + 7f_5).$$

George Boole (1815–1864) was an English mathematician, philosopher and logician. He worked in the fields of differential equations and algebraic logic and is regarded as a founder of the field of computer science.



Exercise 9.4. Maximum degree of precision.

1. Given two real numbers α_1 and α_2 and two values t_1, t_2 in $[-1, 1]$. For a function $f : [-1, 1] \rightarrow \mathbb{R}$, consider the basic quadrature $J_1(f) := \alpha_1 f(t_1) + \alpha_2 f(t_2)$ to approximate $\int_{-1}^1 f(t) dt$. How do we choose the four numbers α_1, α_2, t_1 and t_2 to obtain the maximum degree of precision? ▷ *Math Hint*⁴ ◁

³ See (9.1).

⁴ Write the equations $J_1(p_i) = \int_{-1}^1 p_i(t) dt$ for $p_i(t) = t^i$ and $i = 0, 1, 2, \dots$

Comment: The computed values of the α_i 's give the Gauss-Legendre rule (see Exercise 9.6). ☺

2. Now we consider the quadrature $J_2(f) := \beta_1 f(-1) + \beta_2 f(0) + \beta_3 f(1)$. Compute the values of the β_i s to obtain degree of precision 2. Is 2 the maximum degree of precision? ▷ Math Hint⁵ ◁

Comment: The computed values of the β_i 's give the Simpson rule, see the review and Exercise 9.2 ☺

Exercise 9.5. Also in dimension 2.

Given a triangle T in the plane with surface area $A(T)$ and vertices $P_i = (x_i, y_i)$, $i = 1, 2, 3$. Can we choose a point $\hat{P} = (\hat{x}, \hat{y})$ such that

$$\iint_T f(x, y) dx dy = A(T)f(\hat{P}) \quad (9.3)$$

for any linear function $f(x, y)$ defined on the triangle?

Comment: Those who are familiar with triangular barycentric coordinates can go directly to Part 3. ☺

- Review:** If the vertices of T are ordered counterclockwise then $A(T) = \frac{1}{2} \det M_T > 0$, where

$$M_T := \begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}. \quad (9.4)$$

The affine function $L(x, y) := [x_1, y_1]^T + M_T[x, y]^T$ maps the unit triangle T_0 with vertices $\{(0, 0), (1, 0), (0, 1)\}$ in a one-to-one fashion onto T . ♣

1. Consider first (9.3) for $T = T_0$ ▷ Math Hint⁶ ◁
2. Treat the general case using the mapping L in the review.
3. Treat the general case using barycentric coordinates (u, v, w) on T .

▷ Math Hint⁷ ◁

9.1.2 Weighted Quadrature Rules

Review:

1. weighted quadrature rule:

$$J(f) = \sum_{i=1}^n w_i f(x_i) \approx I(f) = \int_a^b f(x) w(x) dx,$$

⁵ Once the β_i s are known, try the quadrature with $p(t) = t^j$ for $j = 3, 4, \dots$

⁶ In this case we have $\iint_{T_0} f(x, y) dx dy = \int_0^1 \int_0^{1-y} f(x, y) dx dy$.

⁷ If P_1, P_2, P_3 are the vertices of T , for $u + v + w = 1$ and $u, v, w \geq 0$, then $f(uP_1 + vP_2 + wP_3) = uf(P_1) + vf(P_2) + wf(P_3)$.

where $w(x)$ is a positive function on (a, b) called the **weight function** and $x_1 < x_2 < \dots < x_n$. Recall that $\Pi(x) = \prod_{i=1}^n (x - x_i) \in \mathbb{P}_n$ is the corresponding nodal polynomial and ℓ_1, \dots, ℓ_n are the polynomials of the Lagrange basis for \mathbb{P}_{n-1} .

2. To every weight function w there corresponds an inner product on $\mathcal{C}([a, b], \mathbb{R})$ defined by

$$\langle f, g \rangle = \int_a^b f(x)g(x)w(x) dx.$$

We say that a sequence of polynomials $(\varphi_i)_{i \geq 0}$, where $\varphi_i \in \mathbb{P}_i$ are **orthogonal polynomials** with respect to this inner product if $\langle \varphi_i, \varphi_j \rangle = 0$ if $i \neq j$. They are **orthonormal polynomials** if in addition $\langle \varphi_i, \varphi_i \rangle = 1$ for all i .

3. The degree of precision of J is $n + k - 1$ if and only if $J(p) = I(p)$ for all $p \in \mathbb{P}_{n-1}$ and $\langle p, \Pi \rangle = 0$ for all polynomials $p \in \mathbb{P}_k$.
4. The maximal degree of precision of an n point formula is $d = 2n - 1$, the weights $w_i = \langle 1, \ell_i \rangle$ are all positive, the nodes x_1, \dots, x_n are the roots of the orthonormal polynomial φ_n and $a \leq x_1 < x_2 < \dots < x_n \leq b$. For $f \in \mathcal{C}^{2n}([a, b])$, the truncation error $R(f) := I(f) - J(f)$ satisfies

$$R(f) = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b \Pi^2(x)w(x) dx = c_n f^{(2n)}(\xi) \quad (9.5)$$



Exercise 9.6. Gauss-Legendre quadrature

Adrien-Marie Legendre (1752–1833) was a French mathematician. Author of *Éléments de géométrie*, he worked on elliptic functions. Legendre transformation is used in classical mechanics and also in thermodynamics. He is the namegiver of the Legendre polynomials, solutions to Legendre's differential equation, which occur frequently in physics and engineering applications. The Moon crater Legendre is also named after him.



1. Show that $\langle f, g \rangle = \int_{-1}^1 f(x)g(x) dx$ defines an inner product on $\mathcal{C}([-1, 1])$ and find the orthonormal polynomials $\varphi_i \in \mathbb{P}_i$ for $i = 0, 1, 2$.
2. Compute the roots of $\varphi_2(x)$.
3. Let $I(f) := \int_{-1}^1 f(t) dt$. Derive the Gauss-Legendre rule:

$$J(f) := f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right). \quad (9.6)$$

4. Show that $R(f) := I(f) - J(f) = \frac{1}{135} f^{(4)}(\xi)$.

©Comment: A generalization to larger n with $w(x) = 1$ and $[a, b] = [-1, 1]$ is given in the following array:

n	x_i	w_i
2	$\pm\sqrt{3}/3$	1
3	0	8/9
	$\pm\sqrt{15}/5$	5/9
4	$\pm(\sqrt{525} - 70\sqrt{30})/35$	$(18 + \sqrt{30})/36$
	$\pm(\sqrt{525} + 70\sqrt{30})/35$	$(18 - \sqrt{30})/36$
5	0	128/225
	$\pm(\sqrt{245} - 14\sqrt{70})/21$	$(322 + 13\sqrt{70})/900$
	$\pm(\sqrt{245} + 14\sqrt{70})/21$	$(322 - 13\sqrt{70})/900$



Exercise 9.7. Inner product and quadrature.

1. For $f, g \in \mathcal{C}([0, 1])$, show that

$$\langle f, g \rangle = \int_0^1 \frac{f(x)g(x)}{\sqrt{x}} dx$$

is well defined.

2. Show that $\langle \cdot, \cdot \rangle$ defines an inner product on $\mathcal{C}([0, 1], \mathbb{R})$.
3. Find an orthonormal basis $\{\varphi_i\}_{i=1,2,3}$ of $\mathbb{P}_2 \subset \mathcal{C}[0, 1]$ such that $\varphi_i \in \mathbb{P}_{i-1}$.
4. Given two real numbers α_1 and α_2 and two values x_1, x_2 in $[0, 1]$, for a function $f : [0, 1] \rightarrow \mathbb{R}$, consider the basic quadrature $J(f) := \alpha_1 f(x_1) + \alpha_2 f(x_2)$ and $R(f) := \left| \int_0^1 \frac{f(x)}{\sqrt{x}} dx - J(f) \right|$. Find $x_1, x_2, \alpha_1, \alpha_2$ such that for any $q \in \mathbb{P}_3$, $R(q) = 0$. ▷ Math Hint⁸ ◁ Approximation values of the α_i s will be computed using MATLAB.
5. For $f \in C^4([0, 1], \mathbb{R})$, prove the bound of the error $R(f) \leq c_2 M_4(f)$ where $M_4(f) = \max_{t \in [0, 1]} |f^{(4)}(t)|$ and give an approximation of c_2 computed with MATLAB.

Exercise 9.8. Best approximation.

1. From the Gauss-Legendre rule (see (9.6) in Exercise 9.6), deduce a quadrature $J_1(f) := \alpha_1 f(y_1) + \alpha_2 f(y_2)$ for an approximation of $I_1(f) := \int_0^1 f(x) dx$ and give the truncation error $R_1(f)$.

⁸ Use the review at the beginning of the section.

2. Show that

$$\langle f, g \rangle = \int_0^1 x^{1/3} f(x)g(x) dx$$

defines an inner product on $\mathcal{C}([0, 1], \mathbb{R})$.

3. Find an orthonormal basis for $\langle \cdot, \cdot \rangle$, $\{\varphi_i\}_{i=0,1}$ of $\mathbb{P}_1 \subset \mathcal{C}[0, 1]$ such that $\varphi_i \in \mathbb{P}_i$ and show that $\varphi(x) := x^2 - \frac{14}{13}x + \frac{14}{65}$ is orthogonal to \mathbb{P}_1 .
4. If x_1 and x_2 are the roots of $\varphi(x)$, find w_1, w_2 such that $J_2(f) := w_1 f(x_1) + w_2 f(x_2)$ is an approximation of $I_2(f) := \int_0^1 t^{1/3} f(t) dt$ with degree of precision 3.
5. Give the truncation error $R_2(f)$.
6. Give the first terms of the Taylor expansion of $h(x) := \left(\frac{\sin x}{x}\right)^{1/3}$, $h(x) = a_0 + a_2 x^2 + a_4 x^4 + \dots$
7. Compute two approximations of $\int_0^1 (\sin x)^{1/3} dx$, one using the series expansion and the other using the quadrature formula $J_2(f)$.
8. Which one gives the smallest error?

9.2 Monte Carlo Method

Review: The Monte Carlo integration method computes an estimate of a multidimensional definite integral of the form

$$I := \int_{a_1}^{b_1} \int_{a_2}^{b_2} \cdots \int_{a_n}^{b_n} f(x_1, x_2, \dots, x_n) dx_1 dx_2 \cdots dx_n = \int_C f(\mathbf{x}) d\mathbf{x},$$

where C is the hypercube

$$C := \{\mathbf{x} \in \mathbb{R}^n; a_1 \leq x_1 \leq b_1, a_2 \leq x_2 \leq b_2, \dots, a_n \leq x_n \leq b_n\}$$

with volume $V := \prod_{i=1}^n (b_i - a_i)$. The algorithm samples points from the integration region to estimate the integral and its error. Suppose that the sample has size N and denote the points in the sample by $\mathbf{x}^1, \dots, \mathbf{x}^N$. Then the estimate for the integral is given by

$$I \simeq Q_N := \frac{V}{N} \sum_{i=1}^N f(\mathbf{x}^i).$$

“Hit or miss” integration is the simplest type of a Monte Carlo method. If $K \subset C \subset \mathbb{R}^n$, the volume vol_K of K can be approximated by using the function

$$f(\mathbf{x}) := \mathbb{1}_K = \begin{cases} 1 & \text{if } \mathbf{x} \in K \\ 0 & \text{if } \mathbf{x} \notin K \end{cases}$$

in the previous formula. Thus if N_K is the number of x^i in $C \cap K$ then

$$vol_K \approx V \frac{N_K}{N}. \quad (9.7)$$

In the MATLAB `help`, rubric `RandStream`, six available generator algorithms for random numbers are proposed and their properties are given. ♦

Exercise 9.9. Approximation of π .

Since π is the area of a disk of radius 1, we can evaluate $\pi/4$ as the area of K which is the quarter of the disk included in $C = [0, 1]^2$.

Write a program `getpi.m` with input N , the number of points sampled in V and output the approximation of π obtained by $4 \frac{N_K}{N}$ where N_K are the number of sampled points inside K . Test $N = 100, \dots$ ◁ MATLAB Hint⁹ ▷

```
>> getpi
Enter an integer: 100
NK =
    81
Approximation of pi:
ans =
    3.2400
```

Exercise 9.10. Volume of an ellipsoid. Consider for positive real numbers a, b, c the solid ellipsoid

$$K := \left\{ (x, y, z) \in \mathbb{R}^3 : \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \leq 1 \right\}. \quad (9.8)$$

1. Let I denote the interval $[-1, 1]$. Show that K is contained in the hypercube

$$C := \{(au, bv, cw) : (u, v, w) \in C_B\}, \quad C_B := I^3 := I \times I \times I.$$

2. Show that (9.7) takes the form

$$vol_K \approx 8abc \frac{N_B}{N},$$

where N_B is the number of points in C_B sampled from the unit ball

$$B := \{(u, v, w) \in \mathbb{R}^3 : u^2 + v^2 + w^2 \leq 1\}.$$

3. Using the Monte Carlo method, write a MATLAB program that computes an approximation of the volume vol_K of the ellipsoid corresponding to $a = 1$, $b = 2$, and $c = 3$, and adds the computation of $vol/8$.

```
>> volK
Enter an integer: 100
```

⁹ Use the function `nnz.m`.

```

Approximation of volume of K:
vol_k =
    24
Approximation of pi:
ans =
    3

```

⊕Comment: Since the volume of the solid ellipsoid K is $\frac{4}{3}\pi abc = 8\pi$ it follows that $vol/8$ is an approximation to π . ⊕

4. Use the Monte Carlo method to obtain an approximation of the volume $k := K_1 \cap K_2$ and $K := K_1 \cup K_2$ where K_1 and K_2 are given by (9.8) with $a = 1$, $b = 2$, $c = 3$ for K_1 and $a = 4$, $b = 3$, $c = 2$ for K_2 . ▷Math Hint¹⁰◀

```

>> volkK
Enter an integer: 100
Approximation of volume of k:
vol_k =
    23.6800

```

9.3 Trapezoidal Rule

Exercise 9.11. Elementary Quadrature on $[a, b]$. The results given below were already obtained in Exercise 9.1 but in this exercise, we propose (new) elementary proofs.

Let $f \in \mathcal{C}([a, b], \mathbb{R})$. We are looking for approximations of $I(f) := \int_a^b f(t) dt$ and one of them is the trapezoidal rule given by $j(f) := (b - a) \frac{f(a) + f(b)}{2}$.

1. What is the exact degree of precision of the rule?
2. Show that for $f \in C^1([a, b])$, then $I(f) = j(f) - \int_a^b f'(t) \left(t - \frac{a+b}{2} \right) dt$. ▷Math Hint¹¹◀
3. Show that $\bar{j}(f) := \int_a^b f' \left(\frac{a+b}{2} \right) \left(t - \frac{a+b}{2} \right) dt = 0$.
4. From the two previous results, deduce that for $f \in C^2[a, b]$, the truncation error, $r(f) := I(f) - j(f)$, satisfies

$$|r(f)| \leq \frac{1}{12} (b-a)^3 M_2 \text{ where } M_2 = \max_{t \in [a,b]} |f^{(2)}(t)|. \quad (9.9)$$

¹⁰ Find an hypercube v containing k and one hypercube V containing K .

¹¹ Use integration by parts and the primitive function $t - \frac{a+b}{2}$ of the constant 1.

▷ *Math Hint*¹² ◁

Exercise 9.12. Composite rules

For $n \in \mathbb{N} \setminus \{0\}$, we define $h = (b - a)/n$ and for $k = 1, \dots, n + 1$, $x_k = a + (k - 1)h$. On each interval, $[x_k, x_{k+1}]$, $\int_{x_k}^{x_{k+1}} f(t) dt$ is approximated by the previous basic quadrature, then the sum is computed.

1. Deduce $J(f, n)$, the new approximation of $I(f)$.
2. Bound $|R(f, n)|$ where $R(f, n) := I(f) - J(f, n)$.

Exercise 9.13. Computation with the trapezoidal rule.

The aim of the exercise is the computation of an approximation of $I(f) = \int_a^b f(t) dt$ using the trapezoidal rule. We use the notation of the previous exercise and the approximation is given by

$$J(f, n) = \frac{h}{2} \left[f(a) + f(b) + 2 \sum_{k=1}^{n-1} f(a + kh) \right] \text{ where } h = \frac{b-a}{n}. \quad (9.10)$$

1. Write a file `f.m` that compute the values of the function f , for example $f(x) = \exp(x)$.

```
>> f([0.5 1])
ans =
    1.6487    2.7183
```

2. Write a function `trap.m` that computes the approximation $J(f, n)$ Test with `trap(a,b,filef,n)`, $a = 0$, $b = 1$, `filef = 'f'`, $n = 10$. ◁ MATLAB Hint¹³ ▷

```
>> trap(0,1,'f',10)
ans =
    1.7197
```

3. Write a program to study the error $|I(f) - J(f, n)|$ when modifying n . Start with an array `arrn = [1, 2, 5, 7, 10, 15, 20, 35, 50, 75, 100]` to compute the corresponding array `arrerr`, then plot $\log(arrerr)$ depending on $\log(arrn)$. Test $a = 0$, $b = 1$, `filef = 'f'`. Save as `traperror.m`. ◁ MATLAB Hint¹⁴ ▷

©Comment: This numerical computation will show that error $\simeq C/n^\alpha$ or equivalently $\log(\text{error}) \simeq \log(C) - \alpha \log(n)$. ©

```
>> traperror
Coefficient of the regression line
u =
    -1.9976    -1.9518
```

¹² Write $I(f) - j(f) = \bar{j}(f) + I(f) - j(f)$ and use the mean value theorem.

¹³ Use `feval` to evaluate f at a point.

¹⁴ `polyfit(log(tabn), log(taberr), 1)` gives the two coefficients of the regression line.

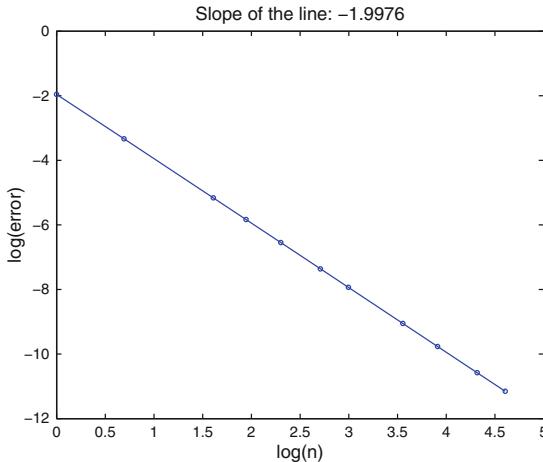


Fig. 9.1 Error in the trapezoidal quadrature

See also Fig. 9.1.

Comment: The slope of the line is approximatively -2 (value: -1.9976), which gives error $\simeq C/n^2$. This was the theoretical bound in the previous exercise. ☺

Exercise 9.14. Application to an integral equation

The goal of the exercise is the computation at sampled points of an approximation \mathbf{V} of the solution u of the integral equation

$$u(x) = \int_a^b K(x, t)u(t)dt + f(x), \quad x \in [a, b], \quad (9.11)$$

where the functions K and f are given. The function K is called the kernel of the equation. An example of a such equation is Love's equation used in electrostatic:

$$u(x) = \frac{1}{\pi} \int_{-1}^1 \frac{1}{1 + (x-t)^2} u(t)dt + 1.$$

The approximation of the integral is computed using the composite trapezoidal rule. We recall that for a given positive integer n , a uniform partition of $[a, b]$ is defined by $h = (b-a)/n$ and $x_i = a + (i-1)h$ for $i = 1$ to $n+1$. Then $I(\varphi) := \int_a^b \varphi(t) dt$ is approximated by $J(\varphi, n) := \frac{h}{2} \left(\varphi(x_1) + 2 \sum_{j=2}^n \varphi(x_j) + \varphi(x_{n+1}) \right)$.

1. Show that if v_j is the approximation of $u(x_j)$, the approximate problem can be written

$$(\mathbf{I} - \frac{h}{2} \mathbf{A}_n) \mathbf{V} = \mathbf{F}$$

where \mathbf{I} is the identity matrix in $\mathbb{R}^{(n+1) \times (n+1)}$,

$$\mathbf{A}_n = \begin{bmatrix} K(x_1, x_1) & 2K(x_1, x_2) & \dots & 2K(x_1, x_n) & K(x_1, x_{n+1}) \\ K(x_2, x_1) & 2K(x_2, x_2) & & & \vdots \\ \vdots & \vdots & & \vdots & \vdots \\ K(x_{n+1}, x_1) & 2K(x_{n+1}, x_2) & \dots & 2K(x_{n+1}, x_n) & K(x_{n+1}, x_{n+1}) \end{bmatrix},$$

$\mathbf{V} = [v_1, v_2, \dots, v_{n+1}]^T \in \mathbb{R}^{n+1}$ is the unknown vector,

$$\mathbf{F} = [f(x_1), f(x_2), \dots, f(x_{n+1})]^T.$$

In the following questions, we suppose that the function K satisfies $K(x, t) = k(x - t)$ for some function $k : \mathbb{R} \rightarrow \mathbb{R}$.

2. Write a first program (without any loop) that given a, b and n computes the vector \mathbf{x} , then the matrix $\mathbf{B}_n \in \mathbb{R}^{(n+1) \times (n+1)}$ defined by

$$B_n = [x_i - x_j] = \begin{bmatrix} 0 & x_1 - x_2 & \dots & x_1 - x_n & x_1 - x_{n+1} \\ x_2 - x_1 & 0 & & \vdots & \vdots \\ \vdots & \vdots & & 0 & x_n - x_{n+1} \\ x_{n+1} - x_1 & x_{n+1} - x_2 & & x_{n+1} - x_n & 0 \end{bmatrix}$$

Save as `inteq1.m`. Test $a = -1, b = 1, n = 3$, display \mathbf{B}_n .

```
>> inteq1
Bn =
    0   -0.6667   -1.3333   -2.0000
  0.6667      0   -0.6667   -1.3333
  1.3333   0.6667      0   -0.6667
  2.0000   1.3333   0.6667      0
```

3. Write two function files that given t , an array of real numbers, compute $k(t)$ and $f(t)$. Save as `k.m` and `f1.m`. Test with `f1([1 2], id)` for `k`.

Example : $k(t) = t^2, f_1(t) = \sin(\pi t) + \frac{4t}{\pi}$.

```
>> k([1 2])
ans =
    1      4
>> f1([1 2])
ans =
    1.2732    2.5465
```

4. Complete `inteq1.m` with the construction of the matrices $\mathbf{C}_n = [k(x_i - x_j)]_{i,j=1,\dots,n+1}$ then \mathbf{A}_n . Save as `inteq2.m`. Test with $a = -1, b = 1, n = 3$, display \mathbf{A}_n .

```
>> inteq2
An =
    0    0.8889    3.5556    4.0000
  0.4444      0    0.8889    1.7778
  1.7778    0.8889      0    0.4444
  4.0000    3.5556    0.8889      0
```

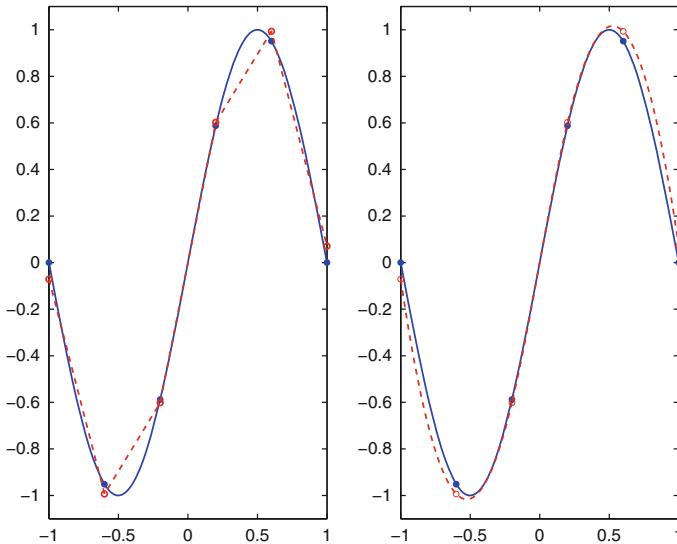


Fig. 9.2 Exact and approximate solutions of the integral equation

5. Complete `inteq2.m` with the construction of \mathbf{F}_n and the solution \mathbf{V} of $(\mathbf{I} - \frac{h}{2} \mathbf{A}_n) \mathbf{V} = \mathbf{F}$. Save as `inteq3.m`. Test : $a = -1$, $b = 1$, $n = 5$, display \mathbf{V} .

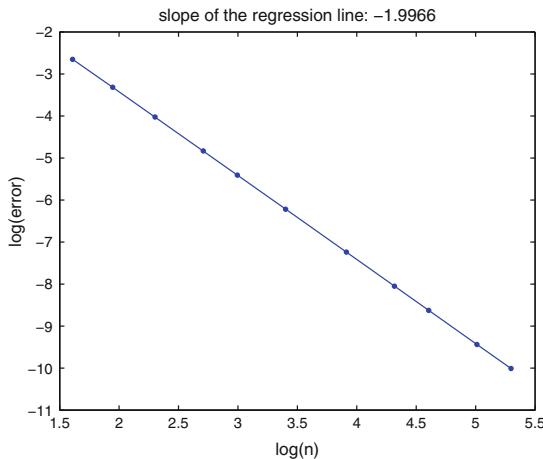
```
>> inteq3
V =
-0.0705
-0.9934
-0.6019
0.6019
0.9934
0.0705
```

6. Compute the error: $err = \|\mathbf{U} - \mathbf{V}\|_\infty = \max_{i=1, \dots, n+1} (|U_i - V_i|)$ where $U_i = u(x_i)$ then plot the piecewise linear curve interpolating the points (x_i, V_i) for $i = 1$ to $n + 1$ and the exact solution which is $u(t) = \sin(\pi t)$. A cubic spline (cf. online help) can be plotted instead of the piecewise linear curve. Save as `inteq4.m`. Test : $a = -1$, $b = 1$, $n = 5$ display err and plots.

```
>> inteq4
err =
0.0705
```

Also see Fig. 9.2.

7. Study the error $\|\mathbf{U} - \mathbf{V}\|_\infty$ for different values of n . Start with an array $arrn = [5, 7, 10, 15, 20, 30, 50, 75, 100, 150, 200]$ to compute the corresponding array $arrerr$, then plot $\log(arrerr)$ depending on $\log(arrn)$. Save as `inteq5.m`. Test : $a = -1$, $b = 1$. Conclusion on the degree of precision.

**Fig. 9.3** Error in the integral equation

```
>> inteq5
Coefficients of the regression line
a =
-1.9966      0.5699
```

Also see Fig. 9.3.

8. Write a new program and the corresponding error study with the composite Simpson quadrature instead of the trapezoidal rule:

$$\int_a^b \varphi(t) dt \simeq \frac{h}{3} \left[\varphi(x_1) + \varphi(x_{N+1}) + 2 \sum_{k=1}^{N'-1} \varphi(x_{2k+1}) + 4 \sum_{k=1}^{N'} \varphi(x_{2k}) \right],$$

where $N = 2N'$ and $h = (b - a)/N$. Save as `inteq6.m` (and `inteq7.m` for the error). Test with $arrn' = [3, 4, 5, 6, 10, 15, 37, 50, 75, 100]$, $arrn = 2 \times arrn'$, $a = -1$, $b = 1$.

```
>> inteq7
Coefficient of the regression line:
a =
-4.0277      1.6790
```

©Comment: The error satisfies $\text{err} \simeq C'n^{-4}$ (slope of the regression line -4.0277). ☺

9.4 Extrapolation

Exercise 9.15. Euler-Maclaurin formula.

Colin Maclaurin (1698–1746) was a Scottish mathematician who made important contributions to geometry and algebra. The Maclaurin series, a special case of the Taylor series, are named after him. Independently from Euler and using the same methods, Maclaurin discovered the Euler-Maclaurin formula.



1. Let $\varphi \in C^1([0, 1])$. Show that

$$\int_0^1 \varphi(t) dt = \frac{1}{2}(\varphi(0) + \varphi(1)) - \int_0^1 (t - 1/2)\varphi'(t) dt.$$

2. Let α_i be the sequence of real numbers and $p_i \in \mathbb{P}_i$ the sequence of polynomials defined by the recursion:

$$\begin{aligned} p_1(t) &= -t + 1/2 \quad , \quad \alpha_1 = 0, \\ p_{i+1}(t) &= \int_0^t (\alpha_i - p_i(u)) du \quad , \quad \alpha_{i+1} = \int_0^1 p_{i+1}(t) dt, \text{ for } i \geq 1. \end{aligned}$$

Prove that for $\varphi \in C^\ell([0, 1])$,

$$\begin{aligned} \int_0^1 \varphi(t) dt &= \frac{1}{2}(\varphi(0) + \varphi(1)) + \sum_{i=1}^{\ell-1} \alpha_i \int_0^1 \varphi^{(i)}(t) dt \\ &\quad + \int_0^1 p_\ell(t) \varphi^{(\ell)}(t) dt. \end{aligned} \tag{9.12}$$

▷ *Math Hint*¹⁵ ◁

3. Show that $p_i(t) = (-1)^i p_i(1-t)$ and find α_{2j+1} . ▷ *Math Hint*¹⁶ ◁
4. We introduce the 1-periodic function $\tilde{p}_i(t) := p_i(t - [t])$ where $[t]$ is the integer part of a real number t . Prove that for k, ℓ positive integers and $\psi \in C^{2\ell}([0, k])$, then

$$\begin{aligned} \int_0^k \psi(t) dt &= \frac{1}{2} \sum_{i=0}^{k-1} (\psi(i) + \psi(i+1)) + \sum_{j=1}^{\ell-1} \alpha_{2j} \int_0^k \psi^{(2j)}(t) dt \\ &\quad + \int_0^k \tilde{p}_{2\ell}(t) \psi^{(2\ell)}(t) dt. \end{aligned} \tag{9.13}$$

5. We use the notations of Exercise 9.12 and $J(f, n)$ defined in (9.10). For $f \in C^{2\ell}([a, b])$, show the Euler Maclaurin formula:

¹⁵ Use an induction and compute $\int_0^1 [p_{\ell+1}(t) \varphi^{(\ell)}(t)]' dt$.

¹⁶ Use induction and differentiation to prove that $p_{i+1}(t) - (-1)^{i+1} p_{i+1}(1-t) = \lambda_{i+1} t + \mu_{i+1}$.

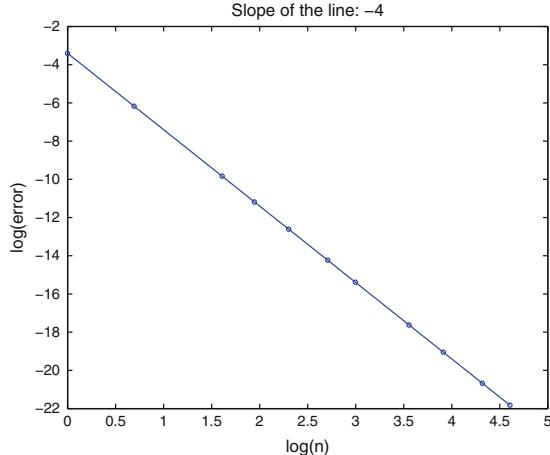


Fig. 9.4 Error in the trapezoidal quadrature for periodic functions

$$\begin{aligned} \int_a^b f(x) dx &= J(f, n) + \sum_{j=1}^{\ell-1} \alpha_{2j} h^{2j} \int_a^b f^{(2j)}(x) dx \\ &\quad + h^{2\ell} \int_a^b \tilde{p}_{2\ell} \left(\frac{x-a}{h} \right) f^{(2\ell)}(x) dx. \end{aligned} \quad (9.14)$$

▷ *Math Hint*¹⁷ ◁

6. From (9.14), deduce that there exists a constant C_ℓ such that if $f \in C^{2\ell}(\mathbb{R})$ is a $(b-a)$ -periodic function, then $\left| \int_a^b f(x) dx - J(f, n) \right| \leq C_\ell h^{2\ell}$ where $h = (b-a)/n$ and compare to the result in Exercise 9.12.
7. Test on the error with a computation with the data of Exercise 9.12, Question 3, except that the function is the 1-periodic function defined on $[0, 1[$ by $f_1(x) = x^2(x-1)^2$, then the π -periodic function f_2 defined on $[0, \pi[$ by $f_2(x) = \sin^2(x)$.

```
>> traperrorf1
Coefficient of the regression line
u =
-4.0000    -3.4012
```

See also Fig. 9.4.

Exercise 9.16. Romberg's method

We start with the composite trapezoidal rule studied in Exercise 9.12. For $n \in \mathbb{N} \setminus \{0\}$, we define $h = (b-a)/n$ and for $k = 1, \dots, n+1$, $x_k = a + (k-1)h$. $I(f) := \int_a^b f(t) dt$ is approximated by

¹⁷ Use $\psi(t) = f(a + th)$.

$$J(f, n) = \frac{h}{2} \left(f(a) + f(b) + 2 \sum_{k=2}^n f(x_k) \right).$$

We have seen in the previous exercise, (9.14), that for a sufficiently smooth function f

$$\int_a^b f(t) dt - J(f, n) = q_{2,2} h^2 + q_{2,4} h^4 + \dots + q_{2,2\ell} h^{2\ell} + o(h^{2\ell}).$$

We define $t_1(f, n) := J(f, n)$, $t_{k+1}(f, n) := \frac{4^k t_k(f, 2n) - t_k(f, n)}{4^k - 1}$ for $k = 1, 2, \dots$

1. Show that $I(f) - t_k(f, n) = 0(h^{2k})$ for $k = 1, 2, \dots, \ell$.

Werner Romberg (1909–2003) was a German mathematician. Being critical to the Nazi regime in Germany, he had to leave his country after his PhD in 1933. He spent some years abroad and came to Norway in 1938. In 1949 Romberg got a position as “dosent” in Trondheim. In his celebrated 1955 paper, Romberg considered numerical approximations of the definite integral. His idea was to accelerate the convergence of the composite trapezoidal rule by means of extrapolation.



2. **Programming:** We use the files `trap.m`, `f.m` of Exercise 9.13 and for a given integer p , we construct the triangular array T :

$T(1, 1) = J(f, 1)$	$T(1, 2)$	$T(1, 3) = J(f, 2^2)$	\dots	$T(1, p)$	$T(1, p+1) = J(f, 2^p)$
$T(2, 1) = \frac{4T(1, 2) - T(1, 1)}{4 - 1}$	$T(2, 2)$		\dots	$T(2, p)$	0
\vdots		\dots		0	\vdots
$T(i+1, 1) = \frac{4^i T(i, 2) - T(i, 1)}{4^i - 1}$	\dots	$T(i+1, p+1-i)$	0		\vdots
\vdots		0			\vdots
$T(p+1, 1) = \frac{4^p T(p, 2) - T(p, 1)}{4^p - 1}$	0	\dots	\dots	\dots	0

- (a) Compute the first row $T(1, :)$. Save as `romb1.m`. Test with $a = 0, b = 1, p = 4$ and $f(x) = \exp(x)$.

```
>> romb1
T =
    1.8591      1.7539      1.7272      1.7205      1.7188
```

- (b) Complete the array T . Save as `romb2.m`. Same Test.

```
>> romb2
T =
    1.8591      1.7539      1.7272      1.7205      1.7188
    1.7189      1.7183      1.7183      1.7183          0
    1.7183      1.7183      1.7183          0          0
    1.7183      1.7183          0          0          0
    1.7183          0          0          0          0
```

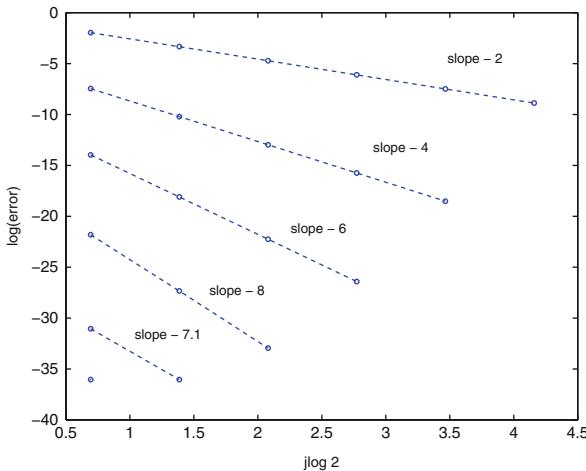


Fig. 9.5 Errors row by row in the Romberg method

3. Complete the previous program with the computation of an array of errors `arrayerr`, then study this error at row i . Plot the curves $\log(\text{arrayerr}(i, j))$ function of $j \times \log(2)$. Save as `romb3.m`. Test with $a = 0, b = 1, p = 5$. See also Fig. 9.5.

```
>> romb3
>> array_err(2,1:5)
ans =
    1.0e-003 *
    0.5793    0.0370    0.0023    0.0001    0.0000
>> array_err(6,1)
ans =
    2.2204e-016
```

©Comment: After row $i = 5$, the computed values are not of the expected precision since we reach the precision of MATLAB. For example $\text{arrayerr}(6, 1) = 2.2204 \times 10^{-16}$. ☺

9.5 A Global Quadrature

Exercise 9.17. Numerical integration

Let f be a continuous function defined on $[a, b]$. We want to approximate $I(f) := \int_a^b f(t) dt$. For a positive integer n , we define $h = (b - a)/n$ and $x_k = a + (k - 1/2)h$ for $k = 1, \dots, n$. The approximation of $I(f)$ is given by

$$\begin{aligned} J(f, n) &= h \left(\frac{1}{9}[f(a) + f(b)] + \frac{7}{8}[f(x_1) + f(x_n)] \right. \\ &\quad \left. + \frac{73}{72}[f(x_2) + f(x_{n-1})] + \sum_{i=3}^{n-2} f(x_i) \right). \end{aligned} \quad (9.15)$$

Comment: This quadrature rule was proposed by our colleague P. Sablonnière as A quadrature formula associated with a univariate quadratic spline quasi-interpolant. in *BIT* **47** (2007) 825-837. ☺

1. Write a function file `f.m` that computes $f(x)$. Test with $f(x) = \exp(x)$.

```
>> f([0.5 1])
ans =
    1.6487      2.7183
```

2. Write a file `approxinteg.m` with the inputs: `filef`, the integer n , the real numbers a , b and output: $J(f, n)$ computed with (9.15). Test the program `approxinteg(filef, n, a, b)` with `filef = 'f'`, $n = 5$, $a = 0$, $b = 1$.

```
>> format long
>> approxinteg('f', 5, 0, 1)
ans =
    1.718273862661467
```

3. Study the error $|I(f) - J(f, n)|$ for different values of n . Start with an array $arrn = 100 : 100 : 800$. Save as `approxintegerr.m`. Test `filef = 'f'`, $a = -1$ and $b = 1$. Conclusion on the degree of precision.
4. A new approximation $J_2(f, n) = \frac{16 * J(f, 2 * n) - J(f, n)}{15}$. Study of the error and conclusion of its degree of precision. Save as `approxintegerr2.m`. Test with $arrn = 80 : 90$ then $arrn = 200 : 210$. Order and comments?

Exercise 9.18. Application to an integral equation

We want to compute an approximation \mathbf{V} at sampled points of the solution u of the integral equation

$$u(x) = \int_a^b K(x, t)u(t)dt + f_1(x), \quad x \in [a, b] \quad (9.16)$$

where the functions K and f_1 are given. See also Exercise 9.14

From a positive integer n , the non-uniform partition on $[a, b]$ is defined by $h = (b - a)/n$, $x_1 = a$, $x_i = a + (i - 3/2)h$ for $i = 2$ to $n + 1$ and $x_{n+2} = b$. For each i from 1 to $n + 2$, in the equation $u(x_i) = \int_a^b K(x_i, t)u(t)dt + f_1(x_i)$, the integral is approximated by (9.15).

1. Show that the approximated problem is

$$(\mathbf{I} - h\mathbf{A}_n)\mathbf{V} = \mathbf{F} \quad (9.17)$$

where \mathbf{I} is the identity matrix in $\mathbb{R}^{(n+2) \times (n+2)}$,

$\mathbf{F} = [f_1(x_1), f_1(x_2), \dots, f_1(x_{n+2})]^T$,

$\mathbf{V} = [V_1, V_2, \dots, V_{n+2}]^T \in \mathbb{R}^{n+2}$ is the vector of unknowns

$\mathbf{A}_n \in \mathbb{R}^{(n+2) \times (n+2)}$ is defined as follow:

$$\text{the 3 first columns are } \begin{bmatrix} \frac{1}{9}K(x_1, x_1) & \frac{7}{8}K(x_1, x_2) & \frac{73}{72}K(x_1, x_3) \\ \frac{1}{9}K(x_2, x_1) & \frac{7}{8}K(x_2, x_2) & \frac{73}{72}K(x_2, x_3) \\ \vdots & \vdots & \vdots \\ \frac{1}{9}K(x_{n+2}, x_1) & \frac{7}{8}K(x_{n+2}, x_2) & \frac{73}{72}K(x_{n+2}, x_3) \end{bmatrix},$$

$$\text{the } n-4 \text{ following columns } \begin{bmatrix} K(x_1, x_4) & \dots & K(x_1, x_{n-1}) \\ K(x_2, x_4) & \dots & K(x_2, x_{n-1}) \\ \vdots & \vdots & \vdots \\ K(x_{n+2}, x_4) & \dots & K(x_{n+2}, x_{n-1}) \end{bmatrix},$$

$$\text{and the 3 last ones } \begin{bmatrix} \frac{73}{72}K(x_1, x_n) & \frac{7}{8}K(x_1, x_{n+1}) & \frac{1}{9}K(x_1, x_{n+2}) \\ \frac{73}{72}K(x_2, x_n) & \frac{7}{8}K(x_2, x_{n+1}) & \frac{1}{9}K(x_2, x_{n+2}) \\ \vdots & \vdots & \vdots \\ \frac{73}{72}K(x_{n+2}, x_n) & \frac{7}{8}K(x_{n+2}, x_{n+1}) & \frac{1}{9}K(x_{n+2}, x_{n+2}) \end{bmatrix}.$$

2. Write a first program that computes the vector $\mathbf{x} = [x_1, \dots, x_{n+2}]^T$ then the

$$\text{matrix } \mathbf{X} = \begin{bmatrix} x_1 & x_1 & \dots & x_1 & x_1 \\ x_2 & x_2 & \dots & x_2 & x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n+2} & x_{n+2} & \dots & x_{n+2} & x_{n+2} \end{bmatrix} \in \mathbb{R}^{(n+2) \times (n+2)} \text{ from given}$$

values of a, b and n . Save as `integeqnew1.m`. Test with $a = -1, b = 1, n = 3$.

```
>> integeqnew1
x =
-1.0000
-0.6667
0
0.6667
1.0000
X =
-1.0000    -1.0000    -1.0000    -1.0000    -1.0000
-0.6667    -0.6667    -0.6667    -0.6667    -0.6667
0          0          0          0          0
0.6667    0.6667    0.6667    0.6667    0.6667
1.0000    1.0000    1.0000    1.0000    1.0000
```

3. Create a function file `K.m` where $K(x, y) = (x - y)^2$. Note that x and y can be arrays, but of the same dimension.

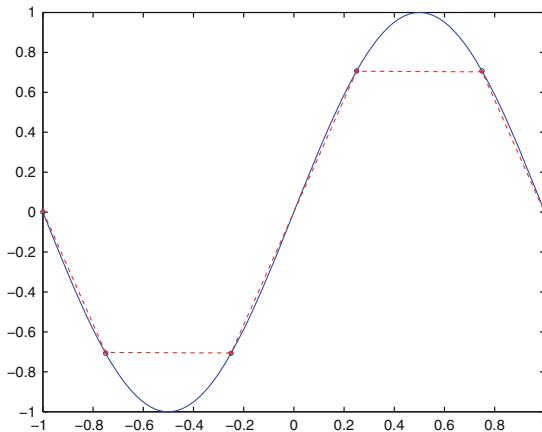


Fig. 9.6 Exact and approximate solutions for $n = 4$

```
>> K([0 1;1 1],[-1 1;2 -3])
ans =
    1      0
    1     16
```

4. Complete `integeqnew1.m` by the computation of $\mathbf{B}_n = [K(x_i, x_j)]_{i,j=1}^{n+2}$ and \mathbf{A}_n . Save as `integeqnew2.m`. Test with $a = -1$, $b = 1$, $n = 4$. Print $\text{An}(2, 3 : 5)$.

```
>> integeqnew2
>> An(2,3:5)
ans =
    0.2535    1.0139    1.9688
```

5. Complete the previous program with the computation of \mathbf{F} with $f_1(t) = \sin(\pi t) + \frac{4t}{\pi}$, then solve the system (9.17) to compute the vector \mathbf{V} . Plot the approximate solution \mathbf{V} and the exact one which is $u(t) = \sin(\pi t)$. Compute the error $\text{err} = \max |V_i - u(x_i)|$. Save as `integeqnew3.m`. Same test.

```
>> integeqnew3
n =
    4
err =
    0.0057
```

See also Fig. 9.6

6. Study the error when n is modified. Save as `integeqnew4.m`. Evaluate the degree of precision of the error. Test with $\text{arrn} = 100 : 110$, $a = -1$, $b = 1$.

9.6 Solutions

9.6.1 Basic Quadrature Rules

Solution of Exercise 9.1

- With the notations of Chap. 6, Exercise 6.3, since $p_n(t) = \sum_{j=1}^n f(x_j)\ell_j(t)$, we deduce that $J(f) = \int_a^b p_n(t) dt = \sum_{j=1}^n w_j f(x_j)$ with $w_j = \int_a^b \ell_j(t) dt$.
- If $q \in \mathbb{P}_{n-1}$ then $q(x) = \sum_{j=1}^n q(x_j)\ell_j(x)$ since $\{\ell_1, \dots, \ell_n\}$ is a basis for \mathbb{P}_{n-1} and $\ell_j(x_i) = \delta_{ij}$. Integrating this identity gives $I(q) = J(q)$.
- Taking absolute values in (9.1)

$$\begin{aligned}|R(f)| &= \left| \frac{1}{n!} \int_a^b f^{(n)}(\xi_x) \Pi(x) dx \right| \leq \frac{1}{n!} \int_a^b |f^{(n)}(\xi_x)| |\Pi(x)| dx \\ &\leq \frac{1}{n!} \max_{t \in [a,b]} |f^{(n)}(t)| \int_a^b |\Pi(t)| dt,\end{aligned}$$

since $\xi_x \in [a, b]$.

Solution of Exercise 9.2

- We have $\ell_1(t) = 1$ when $n = 1$ and so $\int_{-1}^1 \ell_1(t) dt = 2$. Since $x_1 = 0$, we obtain $J_{MP}(f) = 2f(0)$. Now $R_{MP}(1) = 0 = R_{MP}(t)$, but $R_{MP}(t^2) = 2/3$. Thus the exact degree of precision is 1. By (9.2)

$$|R_{MP}(f)| \leq \max_{t \in [-1,1]} |f'(t)| \int_{-1}^1 |t| dt = \max_{t \in [-1,1]} |f'(t)|,$$

which is not optimal since the exact degree of precision is 1.

- With $n = 2$, $x_1 = -1$ and $x_2 = 1$, we obtain $\ell_1(t) = \frac{1-t}{2}$ and $\ell_2(t) = \frac{1+t}{2}$. Then $\int_{-1}^1 \ell_i(t) dt = 1$ and $J_T(f) = f(-1) + f(1)$. Now $R_T(1) = 0 = R_T(t)$, but $R_T(t^2) = 2/3 - 2 = -4/3$. Thus the exact degree of precision is 1. Moreover,

$$|R_T(f)| \leq \frac{1}{2} \max_{t \in [-1,1]} |f''(t)| \int_{-1}^1 |t^2 - 1| dt = \frac{2}{3} \max_{t \in [-1,1]} |f''(t)|$$

which is optimal since for $f(t) = t^2$, $|R_T(f)| = \frac{2}{3} \max_{t \in [-1,1]} |f''(t)|$.

3. With $n = 3$, $x_1 = -1$, $x_2 = 0$ and $x_3 = 1$, we obtain $\ell_1(t) = \frac{(t-1)t}{2}$, $\ell_2(t) = 1-t^2$ and $\ell_3(t) = \frac{(t+1)t}{2}$. Then $\int_{-1}^1 \ell_1(t) dt = \int_{-1}^1 \ell_3(t) dt = 1/3$ and $\int_{-1}^1 \ell_2(t) dt = 4/3$ thus $J_S(f) = \frac{1}{3}(f(-1) + 4f(0) + f(1))$. We find $R_S(t^i) = 0$ for $i = 0, \dots, 3$, but $R_T(t^4) = 2/5 - 2/3 \neq 0$. Thus the exact degree of precision is 3. Now

$$|R_S(f)| \leq \max_{t \in [-1, 1]} |f^{(3)}(t)| \int_{-1}^1 |t(t^2 - 1)| dt = \frac{1}{2} \max_{t \in [-1, 1]} |f^{(3)}(t)|$$

which is not optimal.

4. $2J_{MP}(f) + J_T(f) = 4f(0) + f(-1) + f(1) = 3J_S(f)$.

Solution of Exercise 9.3

1. Simpson's 3/8 rule: We find $R_{S38}(t^i) = 0$ for $i = 0, \dots, 3$, but $R_{S38}(t^4) = -1/270 \neq 0$. Thus the exact degree of precision is $d = 3$. Choose the function $f(t) = t^4$ on $[-1, 1]$. Then $f^{(4)}(t) = 4!$ and the error is $R_{S38}(f) = c_{S38}4!$ for some constant c_{S38} .

```
T=0:1/3:1;
n=4;
F=T.^n;
I=1/(n+1);
J=1/8*(F(1)+F(4)+3*(F(2)+F(3)));
C=abs(I-J)/prod(1:n)
C-1/6480
```

©Comment: The error term for Simpson's 3/8 rule can be written $R_{S38}(f) = \frac{(b-a)^5}{6480} f^{(4)}(\xi)$ for some $\xi \in [a, b]$. ☺

2. Boole's rule: We run the following program with $f(t) := t^n$ on $[0, 1]$ and increasing values of n . Before $n = 6$ the error is 0 up to the rounding errors so that the degree of precision is at least 5. For $n = 6$, we obtain a non zero result, so that the exact degree of precision is 5 and the constant C_B is computed.

```
T=0:1/4:1;
n=6;
F=T.^n;
I=1/(n+1);
J=1/(90)*(7*(F(1)+F(5))+32*(F(2)+F(4))+12*F(3));
C=abs(I-J)/prod(1:n)
C-1/1935360
```

©Comment: Note the use of `prod(1:n)` to compute $n!$. The MATLAB function `factorial(n)` can also be used.

The error term for Boole's rule is $R_B(f) = \frac{(b-a)^7}{1935360} f^{(6)}(\xi)$ for some $\xi \in [a, b]$. ☺

Solution of Exercise 9.4

1. If $I(f) = \int_{-1}^1 f(t) dt$. Let us define $p_i(t) = t^i, i = 0, 1, \dots$. Suppose that

$$(Eq_j) : I(p_j) = J_1(p_j), j = 0, 1, 2, 3.$$

Then

with (Eq_1) , we obtain $(Eq'_1) : \alpha_1 t_1 = -\alpha_2 t_2$,

with (Eq_2) and (Eq'_1) , we deduce $(Eq'_2) : 2/3 = \alpha_1 t_1(t_1 - t_2)$,

with (Eq_3) and the previous results, we get $0 = \alpha_1 t_1(t_1 - t_2)(t_1 + t_2)$ then $(Eq'_3) : 0 = 2/3(t_1 + t_2)$.

We deduce that $t_1 = -t_2$ and with (Eq'_2) , we obtain $t_1 \neq 0$ so that with (Eq'_1) , we get $\alpha_1 = \alpha_2$. From (Eq_0) , $\alpha_1 = \alpha_2 = 1$ and finally, with (Eq'_2) , $t_1^2 = 1/3$. We can choose $t_1 = -1/\sqrt{3}$ and $t_2 = 1/\sqrt{3}$ up to a permutation of t_1 and t_2 .

Conversely, for $\alpha_1 = \alpha_2 = 1$ and $t_2 = -t_1 = 1/\sqrt{3}$, we obtain $I(p_j) = J_1(p_j)$ for $j = 0, \dots, 3$ but $I(p_4) = 2/5 \neq J_1(p_4) = 2/9$. We deduce that the maximum degree of precision is 3.

2. We use the same method testing with the $p_i(t) = t^i, i = 0, 1, \dots$. If $J_2(p_j) = I(p_j)$ for $j = 0, 1, 2$ then

$$\beta_1 + \beta_2 + \beta_3 = 2, -\beta_1 + \beta_3 = 0, \beta_1 + \beta_3 = 2/3.$$

We deduce that $\beta_1 = \beta_3 = 1/3$, and then $\beta_2 = 4/3$. In this case $J_2(p_3) = I(p_3)$ and $J_2(p_4) \neq I(p_4)$, hence the exact degree of precision is 3.

Solution of Exercise 9.5

1. We find $\iint_{T_0} 1 dx dy = 1/2 = A(T_0)$ and $\iint_{T_0} x dx dy = 1/6 = \iint_{T_0} y dx dy$. Thus (9.3) is exact for constants for any choice of $\hat{P} = (\hat{x}, \hat{y})$. Taking $f(x, y) = x$ and $f(x, y) = y$ in (9.3) give the equations $\frac{1}{6} = \frac{1}{2}\hat{x}$ and $\frac{1}{6} = \frac{1}{2}\hat{y}$ with solution $\hat{x} = \hat{y} = 1/3$.
2. Consider the affine transformation $L(x, y) := [x_1, y_1]^T + M_T[x, y]^T$, where M_T is given by (9.4), mapping T_0 in a one-to-one fashion onto T . We have $L(\frac{1}{3}, \frac{1}{3}) = \hat{P} := (P_1 + P_2 + P_3)/3$, the **barycenter** of T . For any linear function f , by a change of variable in the double integral,

$$\begin{aligned} \iint_T f(x, y) dx dy &= |\det M_T| \iint_{T_0} f(L(x, y)) dx dy \\ &= 2A(T)A(T_0)f(L(\frac{1}{3}, \frac{1}{3})) = A(T)f(\hat{P}). \end{aligned}$$

3. Let P_1, P_2, P_3 be the vertices of the triangle T . For $u_1 + u_2 + u_3 = 1$ and $u_1, u_2, u_3 \geq 0$, the points $u_1 P_1 + u_2 P_2 + u_3 P_3$ describe the whole triangle in terms of barycentric coordinates on T . Since f is a linear function, $f(u_1 P_1 + u_2 P_2 + u_3 P_3) = u_1 f(P_1) + u_2 f(P_2) + u_3 f(P_3)$. Thus

$$\begin{aligned}\iint_T f(x, y) dx dy &= \sum_{i=1}^3 f(P_i) \iint_T u_i dx dy \\ &= A(T) \frac{f(P_1) + f(P_2) + f(P_3)}{3} = A(T) f\left(\frac{P_1 + P_2 + P_3}{3}\right).\end{aligned}$$

With $\hat{P} = \frac{P_1 + P_2 + P_3}{3}$ we obtain the result $\iint_T f(x, y) dx dy = A(T) f(\hat{P})$.

Solution of Exercise 9.6

1. $\langle \cdot, \cdot \rangle$ is bilinear and symmetric. Also $\langle f, f \rangle = \int_{-1}^1 f(x)^2 dx \geq 0$. If $\langle f, f \rangle = 0$ then the integral of the non-negative continuous function $f(x)^2$ is 0, hence f^2 is the zero function on $[-1, 1]$ and so is f . We deduce that $\langle \cdot, \cdot \rangle$ is an inner product on $C([-1, 1], \mathbb{R})$.

(a) $\varphi_0(x) = a_0$ and $\int_{-1}^1 \varphi_0(x)^2 dx = 1$ so that $2a_0^2 = 1$ and we can choose $\varphi_0(x) = 1/\sqrt{2}$.

(b) $\varphi_1(x) = a_1x + b_1$, $\int_{-1}^1 \varphi_0(x)\varphi_1(x) dx = 0$ gives $b_1 = 0$ and the choice $a_1 = \sqrt{\frac{3}{2}}$ implies $\int_{-1}^1 \varphi_1(x)^2 dx = 1$.

(c) With $\varphi_2(x) = a_2x^2 + b_2x + c_2$, we solve $\langle \varphi_2, \varphi_i \rangle = 0$ for $i = 0, 1$ to obtain

$$\begin{cases} 2/3a_2 + 2c_2 = 0 \\ 2/3b_2 = 0 \end{cases} \Leftrightarrow \begin{cases} b_2 = 0 \\ c_2 = -\frac{1}{3}a_2 \end{cases}$$

so that $\varphi_2(x) = a_2(x^2 - 1/3)$.

Now $a_2 = \frac{3}{2}\sqrt{\frac{5}{2}} \Leftrightarrow \int_{-1}^1 \varphi_2(x)^2 dx = 1$.

2. The roots of $\varphi_2(x)$ are $x_1 = -\frac{1}{\sqrt{3}}$ and $x_2 = \frac{1}{\sqrt{3}}$.

3. Let $I(f) := \int_{-1}^1 f(t) dt$. We fix $n = 2$. Following the review, we compute the Lagrange basis: $\ell_1(x) = \frac{x - x_2}{x_1 - x_2}$ and $\ell_2(x) = \frac{x - x_1}{x_2 - x_1}$, then the weights $w_i = \langle 1, \ell_i \rangle$. We obtain $w_1 = w_2 = 1$ so that

$$J(f) := f\left(-\frac{1}{\sqrt{3}}\right) + f\left(\frac{1}{\sqrt{3}}\right).$$

4. The degree of precision is 3 and for $f \in \mathcal{C}^4([-1, 1])$, $R(f) := I(f) - J(f)$ satisfies $R(f) = \frac{f^{(4)}(\xi)}{4!} \int_{-1}^1 \Pi^2(x) dx = c_2 f^{(4)}(\xi)$ where $\Pi(x) = (x - x_1)(x - x_2)$. A computation gives $\frac{1}{4!} \int_{-1}^1 (x^2 - 1/3)^2 dx = \frac{1}{135}$ so that $R(f) = \frac{1}{135} f^{(4)}(\xi)$.

Solution of Exercise 9.7

1. Let $f, g \in \mathcal{C}([0, 1], \mathbb{R})$ then $\frac{f(x)g(x)}{\sqrt{x}} \in \mathcal{C}(0, 1]$ and the function $|f(x)g(x)|$ is bounded on $[0, 1]$ by M . Thus $\left| \frac{f(x)g(x)}{\sqrt{x}} \right| \leq \frac{M}{\sqrt{x}}$. The integral $\int_0^1 \frac{M}{\sqrt{x}} dx$ is convergent and so is the integral $\int_0^1 \frac{f(x)g(x)}{\sqrt{x}} dx := \langle f, g \rangle$.
 2. Clearly $\langle \cdot, \cdot \rangle$ is symmetric and bilinear. Also for any function f , we have $\langle f, f \rangle \geq 0$. Finally, if $\langle f, f \rangle = 0$, then the non negative continuous function $h(x) := \frac{f(x)^2}{\sqrt{x}}$ defined on $(0, 1]$ vanishes on the interval $(0, 1]$ since its integral is 0. But then f is zero. By continuity of f at 0, we also have $f(0) = 0$. We deduce that f is the zero function on $[0, 1]$ and all the properties defining inner products are satisfied.
 3. (a) $\varphi_0(x) = a_0$ and $\int_0^1 \frac{\varphi_0(x)^2}{\sqrt{x}} dx = 1$ so that $2a_0^2 = 1$ and we can choose $\varphi_0(x) = 1/\sqrt{2}$.
 - (b) $\varphi_1(x) = a_1x + b_1$ and $\int_0^1 \frac{\varphi_0(x)\varphi_1(x)}{\sqrt{x}} dx = 0$, $\int_0^1 \frac{\varphi_1(x)^2}{\sqrt{x}} dx = 1$. So that
- $$\begin{cases} 2/3a_1 + 2b_1 &= 0 \\ 2/5a_1^2 + 4/3a_1b_1 + 2b_1^2 &= 1 \end{cases} \Leftrightarrow \begin{cases} b_1 &= -a_1/3 \\ 8/45a_1^2 &= 1 \end{cases}$$
- and we can choose $a_1 = \frac{3}{2}\sqrt{\frac{5}{2}}$ and $b_1 = -\frac{1}{2}\sqrt{\frac{5}{2}}$ so that $\varphi_1(x) = \frac{1}{2}\sqrt{\frac{5}{2}}(3x - 1)$.

- (c) With $\varphi_2(x) = a_2x^2 + b_2x + c_2$, we write $\langle \varphi_2, \varphi_i \rangle = 0$ for $i = 0, 1$ and obtain

$$\begin{cases} 2/5a_2 + 2/3b_2 + 2c_2 &= 0 \\ 2/7a_2 + 1/3b_2 &= 0 \end{cases} \Leftrightarrow \begin{cases} b_2 &= -6/7a_2 \\ c_2 &= 3/35a_2 \end{cases}$$

so that $\varphi_2(x) = \frac{a_2}{35}(35x^2 - 30x + 3)$. Now

$$\varphi_2(x)^2 = \frac{a_2^2}{35}(35x^2 - 30x + 3)(x^2 + \lambda_1\varphi_1(x) + \lambda_0\varphi_0(x))$$

so that

$$\int_0^1 \frac{\varphi_2(x)^2}{\sqrt{x}} dx = 1 \Leftrightarrow a_2^2(35 \times 2/9 - 30 \times 2/7 + 3 \times 2/5) = 35,$$

and we can choose $a_2 = \frac{3}{16}\sqrt{2}$.

4. The roots of the polynomial $\varphi_2(x)$ are

$$x_1 = \frac{15 - 2\sqrt{30}}{35} \approx 0.1156, \text{ and } x_2 = \frac{15 + 2\sqrt{30}}{35} \approx 0.7416.$$

We define the Lagrange basis $\ell_1(x) = \frac{x - x_2}{x_1 - x_2}$, $\ell_2(x) = \frac{x - x_1}{x_2 - x_1}$ and $w_i =$

$\langle 1, \ell_i \rangle$. Let $J(f) := w_1 f(x_1) + w_2 f(x_2)$ and $R(f) := \left| \int_0^1 \frac{f(x)}{\sqrt{x}} dx - J(f) \right|$.

Then using the review, the degree of precision is 3 so that for any $p \in \mathbb{P}_3$, $R(p) = 0$

```
function y=l1w(x)
y=-7/24*sqrt(30)*(x-(15+2*sqrt(30))/35)./sqrt(x);
function y=l2w(x)
y=7/24*sqrt(30)*(x-(15-2*sqrt(30))/35)./sqrt(x);
```

```
>> format long
>> alpha1=quadl('l1w',0,1),alpha2=quadl('l2w',0,1),
alpha1 =
1.304290801785990
alpha2 =
0.695708559466765
```

Comment: When modifying the tolerance to increase the precision, for example with $\text{alpha1}=quadl('l1w',0,1,1e-12)$ there is a MATLAB message:

Warning: Minimum step size reached; singularity possible. > In quadl at 101. ↴

5. Also, if $f \in C^4([0, 1], \mathbb{R})$, then by (9.5), we obtain $|R(f)| \leq c_2 M_4(f)$ where $M_4(f) = \max_{t \in [0, 1]} |f^{(4)}(t)|$ and $c_2 = \frac{1}{4!} \int_0^1 \frac{[(x - x_1)(x - x_2)]^2}{\sqrt{x}} dx$ which gives $c_2 \approx 4.837711261601187e-004$

```
function y=pi2w(x)
y=[(x-(15-2*sqrt(30))/35).*...
(x-(15+2*sqrt(30))/35)].^2./sqrt(x);
```

```
>> quadl('pi2w',0,1)/24
ans =
4.837711261601187e-004
```

Solution of Exercise 9.8

1. The Gauss-Legendre rule gives $J(g) := g\left(-\frac{\sqrt{3}}{3}\right) + g\left(\frac{\sqrt{3}}{3}\right)$ for an approximation of $\int_{-1}^1 g(t) dt$. If $f \in C[0, 1]$, for $t \in [-1, 1]$, let $g(t) = f\left(\frac{t+1}{2}\right)$. Then

$$\begin{aligned}\int_{-1}^1 g(t) dt &= \int_{-1}^1 f\left(\frac{t+1}{2}\right) dt = 2 \int_0^1 f(x) dx \\ J(g) &= g\left(-\frac{\sqrt{3}}{3}\right) + g\left(\frac{\sqrt{3}}{3}\right) = f\left(\frac{3-\sqrt{3}}{6}\right) + f\left(\frac{3+\sqrt{3}}{6}\right)\end{aligned}$$

and we deduce $J_1(f) = 1/2J(g) = 1/2f(y_1) + 1/2f(y_2)$ with $y_1 = \frac{3-\sqrt{3}}{6}$

and $y_2 = \frac{3+\sqrt{3}}{6}$ for an approximation of $I_1(f) := \int_0^1 f(x) dx$.

The truncation error $\int_{-1}^1 g(t) dt - J(g) = \frac{g^{(4)}(\xi)}{135}$ has been computed in Exercise 9.6, and $R_1(f) = 1/2 \frac{g^{(4)}(\xi)}{135} = \frac{f^{(4)}(\xi_1)}{4320}$.

2. The proof that $\langle f, g \rangle = \int_0^1 x^{1/3} f(x) g(x) dx$ defines an inner product is similar to the proofs of the previous inner products in Exercises 9.6 and 9.7.
3. Idem to compute $\varphi_0(x) = 4/3$, and $\varphi_1(x) = \frac{1}{3}\sqrt{\frac{10}{3}}(4x - 7)$. If $\varphi(x) := x^2 - \frac{14}{13}x + \frac{14}{65}$, then $\langle \varphi_0, \varphi \rangle = \langle \varphi_1, \varphi \rangle = 0$ thus φ is orthogonal to \mathbb{P}_1 .
4. The roots of $\varphi(x)$ are

$$x_1 = \frac{35 - 3\sqrt{35}}{65} \approx 0.2654117, \quad x_2 = \frac{35 + 3\sqrt{35}}{65} \approx 0.8115114.$$

The weights are the solutions of the equations

$$\frac{3}{4} = \int_0^1 x^{1/3} dx = w_1 + w_2, \quad \frac{3}{7} = \int_0^1 x^{4/3} dx = w_1 x_1 + w_2 x_2,$$

and we find

$$w_1 = \frac{3}{392}(49 - \sqrt{35}) \approx 0.329724, \quad w_2 = \frac{3}{392}(49 + \sqrt{35}) \approx 0.420276.$$

The rule defined by $J_2(f) := w_1 f(x_1) + w_2 f(x_2)$, has degree of precision 3.

5. Let $R_2(f) := I_1(f) - J_2(f)$. If $f \in C^4([0, 1])$, then by (9.5), we obtain $R_2(f) \leq c_2 M_4(f)$ where $M_4(f) = \max_{t \in [0, 1]} |f^{(4)}(t)|$ and

$$c_2 = \frac{1}{4!} \int_0^1 x^{1/3} [(x - x_1)(x - x_2)]^2 dx = \frac{81}{540800}.$$

6. From the Taylor expansion of $\sin x = x - x^3/6 + x^5/120 + o(x^5)$, we find $\sin x/x = 1 - x^2/6 + x^4/120 + o(x^4) = (a_0 + a_2 x^2 + a_4 x^4 + o(x^4))^3 = a_0^3 + 3a_0^2 a_2 x^2 + (3a_0 a_2^2 + 3a_0^2 a_4)x^4 + o(x^4)$ giving $a_0 = 1, a_2 = -1/18, a_4 = -1/3240$. Thus the first terms of the Taylor expansion of $h(x) := \left(\frac{\sin x}{x}\right)^{1/3}$ are

$$h(x) = 1 - \frac{1}{18}x^2 - \frac{1}{3240}x^4 + \dots$$

7. Using the first few terms in the series expansion for $h(x)$ we find

$$\int_0^1 (\sin x)^{1/3} dx \simeq \int_0^1 x^{1/3} \left(1 - \frac{1}{18}x^2 - \frac{1}{3240}x^4\right) dx \approx 0.733391.$$

Using now the quadrature rule $J_2(f)$, we find

$$\int_0^1 (\sin x)^{1/3} dx = \int_0^1 x^{1/3} \left(\frac{\sin x}{x}\right)^{1/3} dx \approx 0.733271.$$

8. With $f1(x) := (\sin x)^{1/3}$ and using `quad('f1', 0, 1, 1e-15)` as the “exact” value we find the errors $-1.90561 * 10^{-6}$ and -0.000121989 for the errors in the quadrature formula and the series expansion, respectively. Thus the quadrature formula is the most accurate.

9.6.2 Monte Carlo Method

Solution of Exercise 9.9

`getpi.m`

```
N=input('Enter an integer: ');
X=rand(N,2);
t=X(:,1).^2+X(:,2).^2-1;
NK=nnz(t<= 0)
disp('Approximation of pi:')
4*NK/N
```

Solution of Exercise 9.10

1. If $(x, y, z) \in K$ then $x^2 \leq a^2$ or $-a \leq x \leq a$ or $x \in aI$. Similarly $y \in bI$ and $z \in cI$. It follows that $K \subset C$.

2. Equation (9.7) takes the form $\text{vol}_K \approx \text{vol}_C \frac{N_K}{N}$, where N_K is the number of sample points in $K \cap C$. Now $\text{vol}_C = 2a^2b^2c = 8abc$. Let $(x, y, z) = (au, bv, cw) \in \mathbb{R}^3$. Then $(x, y, z) \in C \iff (u, v, w) \in C_B$, and $(x, y, z) \in K \iff (u, v, w) \in B$. Thus $(x, y, z) \in K \cap C \iff (u, v, w) \in B \cap C_B$ and the result follows.

3. **volK.m**

```
a=1;b=2;c=3;
N=input('Enter an integer: ');
X=rand(N,3);
T1=X(:,1).^2+X(:,2).^2+X(:,3).^2 - 1;
Nk=nnz(T1<= 0);
disp('Approximation of volume of K:');
vol_k=8*a*b*c*Nk/N
disp('Approximation of pi:');
a*b*c*Nk/N
```

4. A hypercube containing $k := K_1 \cap K_2$ is

$$c := \{(x, y, z) \in \mathbb{R}^3; -1 \leq x \leq 1, -2 \leq y \leq 2, -2 \leq z \leq 2\}.$$

To be in k , a point (x, y, z) has to satisfy both $t_1 := x^2 + \frac{y^2}{4} + \frac{z^2}{9} - 1 \leq 0$ and $t_2 := \frac{x^2}{16} + \frac{y^2}{9} + \frac{z^2}{4} - 1 \leq 0$.

A hypercube containing $K := K_1 \cup K_2$ is

$$C := \{(x, y, z) \in \mathbb{R}^3; -4 \leq x \leq 4, -3 \leq y \leq 3, -3 \leq z \leq 3\}.$$

To be in K , a point (x, y, z) has to satisfy one of the inequations $t_1 \leq 0$ or $t_2 \leq 0$.

volK.m

```
a=1;b=2;c=2;
N=input('Enter an integer: ');
X=2*rand(N,3)-1;
X(:,1)=a*X(:,1);
X(:,2)=b*X(:,2);
X(:,3)=c*X(:,3);

T1=X(:,1).^2+X(:,2).^2/4+X(:,3).^2/9 - 1;
T2=X(:,1).^2/16+X(:,2).^2/9+X(:,3).^2/4 - 1;
Nk=nnz((T1<= 0)&(T2<=0));
disp('Approximation of volume of k:');
vol_k=8*a*b*c*Nk/N

%for K: a=4;b=3;c=3;
%NK=nnz(( T1<= 0)|( T2<=0));
```

9.6.3 Trapezoidal Rule

Solution of Exercise 9.11

1. Let $p_i(t) = t^i$ for $i = 0, 1, 2$.

Then $I(p_0) = (b - a) = j(p_0)$, $I(p_1) = \frac{b^2 - a^2}{2} = j(p_1)$, and for $a \neq b$, $I(p_2) \neq j(p_2)$. Therefore, the exact degree of precision is 1.

2. Using an integration by parts, since a primitive function of 1 is $t - \frac{a+b}{2}$, we obtain

$$\begin{aligned} I(f) &= \int_a^b f(t) dt = \left[f(t) \left(t - \frac{a+b}{2} \right) \right]_a^b - \int_a^b f'(t) \left(t - \frac{a+b}{2} \right) dt \\ &= j(f) - \int_a^b f'(t) \left(t - \frac{a+b}{2} \right) dt, \end{aligned}$$

3.

$$\bar{j}(f) = \int_a^b f' \left(\frac{a+b}{2} \right) \left(t - \frac{a+b}{2} \right) dt = f' \left(\frac{a+b}{2} \right) \int_a^b \left(t - \frac{a+b}{2} \right) dt = 0.$$

4. From the previous results

$$\begin{aligned} I(f) - j(f) &= - \int_a^b f'(t) \left(t - \frac{a+b}{2} \right) dt \\ &= \bar{j}(f) - \int_a^b f'(t) \left(t - \frac{a+b}{2} \right) dt \\ &= \int_a^b \left(f' \left(\frac{a+b}{2} \right) - f'(t) \right) \left(t - \frac{a+b}{2} \right) dt \end{aligned}$$

The mean value theorem gives the existence of c_t between $\frac{a+b}{2}$ and t such that $f' \left(\frac{a+b}{2} \right) - f'(t) = \left(\frac{a+b}{2} - t \right) f''(c_t)$ so that $r(f) = - \int_a^b f''(c_t) \left(t - \frac{a+b}{2} \right)^2 dt$. We conclude that

$$|r(f)| \leq M_2 \int_a^b \left(t - \frac{a+b}{2} \right)^2 dt = \frac{1}{12} M_2 (b-a)^3,$$

with $M_2 = \max_{t \in [a,b]} |f^{(2)}(t)|$.

Solution of Exercise 9.12

1. Since $I(f) = \int_a^b f(t) dt = \sum_{k=1}^n \int_{x_k}^{x_{k+1}} f(t) dt$, we can construct an approximation on each interval $[x_k, x_{k+1}]$ and when adding them, we obtain

$$J(f, n) = \sum_{k=1}^n (x_{k+1} - x_k) \frac{f(x_{k+1}) + f(x_k)}{2} = \frac{h}{2} \left[f(a) + f(b) + 2 \sum_{k=2}^n f(x_k) \right].$$

2. Since $\max_{t \in [x_k, x_{k+1}]} |f''(t)| \leq \max_{t \in [a, b]} |f''(t)| =: M_2$, using (9.9) on each interval $[x_k, x_{k+1}]$, we deduce that

$$\begin{aligned} |R(f, n)| &= \left| \sum_{k=1}^n \int_{x_k}^{x_{k+1}} f(t) dt - (x_{k+1} - x_k) \frac{f(x_{k+1}) + f(x_k)}{2} \right| \\ &\leq \sum_{k=1}^n \left| \int_{x_k}^{x_{k+1}} f(t) dt - (x_{k+1} - x_k) \frac{f(x_{k+1}) + f(x_k)}{2} \right| \\ &\leq \sum_{k=1}^n \frac{1}{12} h^3 M_2 = \frac{M_2}{12} (b - a)^3 \frac{1}{n^2}. \end{aligned}$$

Solution of Exercise 9.13

1. `f.m`

```
function y = f(x)
y=exp(x);
```

2. `trap.m`

```
function J = trap(a,b,filef,n)
%composite trapezoidal rule
h=(b-a)/n;
x=a:h:b;
y=feval(filef,x);
J=h/2*(y(1)+y(n+1)+2*sum(y(2:n)));
```

3. `traperror.m`

```
tabn=[1,2,5,7,10,15,20,35,50,75,100];
I=exp(1)-1;
for i=1:length(tabn)
    J=trap(0,1,'f',tabn(i));
    taberr(i)=abs(J-I);
end;
plot(log(tabn),log(taberr),'o-')
disp('Coefficient of the regression line')
u=polyfit(log(tabn),log(taberr),1)
xlabel('log(n)')
ylabel('log(error)')
title(['Slope of the line: ',num2str(u(1))])
```

Solution of Exercise 9.14

1. When we apply (9.11) at x_i and the trapezoidal rule, we obtain that

$$\begin{aligned} u(x_i) &= \int_a^b K(x_i, t)u(t)dt + f(x_i) \\ &\simeq \frac{h}{2} \left(K(x_i, x_1)u(x_1) + 2 \sum_{j=2}^n K(x_i, x_j)u(x_j) + K(x_i, x_{n+1})u(x_{n+1}) \right) \\ &\quad + f(x_i) \end{aligned}$$

from which we deduce

$$v_i = \frac{h}{2} \left(K(x_i, x_1)v_1 + 2 \sum_{j=2}^n K(x_i, x_j)v_j + K(x_i, x_{n+1})v_{n+1} \right) + f(x_i).$$

When collecting the previous equations for $i = 1, \dots, n+1$, we get

$$(\mathbf{I} - \frac{h}{2}\mathbf{A}_n)\mathbf{V} = \mathbf{F}.$$

2. `inteq1.m`

```
n=3
a=-1,b=1
h=(b-a)/n
X=(a:h:b)';
BB=X*ones(1,n+1);
Bn=BB-BB';
```

3. `k.m`

```
function y=k(t)
y=t.*t;
```

`f1.m`

```
function y=f1(t)
y=sin(pi*t)+4*t/pi;
```

4.5.6. `ineq4.m`

```
n=5
a=-1,b=1
h=(b-a)/n
X=(a:h:b)';
BB=X*ones(1,n+1);
Bn=BB-BB';
An=k(Bn);
An(:,2:n)=2*An(:,2:n);
Fn=f1(X);
V=(eye(n+1)-h/2*An)\Fn
```

```

U=u(X);
err=norm(U-V, inf)
t=a:(b-a)/500:b;
subplot(1,2,1)
plot(t,u(t), 'b', X,U, 'ob', X,V, 'o--r')
axis([-1 1 -1.1 1.1])
subplot(1,2,2)
plot(t,u(t), 'b', X,U, 'ob', t, spline(X,V,t), '—r', X,V, 'or')
axis([-1 1 -1.1 1.1])

```

7. Error with different n : inteq5.m

```

arrn=[5,7,10,15,20,30,50,75,100,150,200];
a=-1,b=1
for i=1:length(arrn)
    n=arrn(i);
    h=(b-a)/n;
    X=(a:h:b)';
    BB=X*ones(1,n+1);
    Bn=BB-BB';
    An=k(Bn);
    An(:,2:n)=2*An(:,2:n);
    Fn=f1(X);
    V=(eye(n+1)-h/2*An)\Fn
    U=u(X);
    arrerr(i)=norm(U-V, inf)
end
plot(log(arrn), log(arrerr), 'o-')
disp('Coefficient of the regression line: ')
a=polyfit(log(arrn), log(arrerr), 1)
title(['slope of the regression line: ', ...
    num2str(a(1))], 'FontSize', 16)
xlabel('log(n)', 'FontSize', 16)
ylabel('log(error)', 'FontSize', 16)

```

8. Simpson's quadrature: inteq7.m

```

arrnp=[3,4,5,6,10,15,37,50,75,100];
arrn=2*arrnp
a=-1,b=1
for i=1:length(arrnp)
    n=arrn(i);
    h=(b-a)/n;
    X=(a:h:b)';
    BB=X*ones(1,n+1);
    Bn=BB-BB';
    An=k(Bn);
    An(:,2:2:n)=4*An(:,2:2:n);
    An(:,3:2:n-1)=2*An(:,3:2:n-1);
    Fn=f1(X);
    V=(eye(n+1)-h/3*An)\Fn
    U=u(X);
    arrerr(i)=norm(U-V, inf)
end
plot(log(arrn), log(arrerr), 'o-')

```

```

| disp('Coefficient of the regression line: ')
| a=polyfit(log(arrr),log(arerr),1)
| title(['slope of the regression line: ',...
|       num2str(a(1))], 'FontSize',16)
| xlabel('log(n)', 'FontSize',16)
| ylabel('log(error)', 'FontSize',16)

```

9.6.4 Extrapolation

Solution of Exercise 9.15

1. Using an integration by parts, since a primitive function of 1 is $t - 1/2$, we obtain

$$\begin{aligned}\int_0^1 \varphi(t) dt &= [\varphi(t)(t - 1/2)]_0^1 - \int_0^1 \varphi'(t)(t - 1/2) dt \\ &= \frac{1}{2}(\varphi(0) + \varphi(1)) - \int_0^1 (t - 1/2)\varphi'(t) dt.\end{aligned}$$

2. We prove the formula (9.12) by induction.

For $\ell = 1$, this is the previous result and $p_1 \in \mathbb{P}_1$.

By its definition, $p_{\ell+1} \in \mathbb{P}_{\ell+1}$ as soon as $p_\ell \in \mathbb{P}_\ell$. Also, since $p_{\ell+1}(0) = p_{\ell+1}(1) = 0$, for $\ell \geq 0$, if $\varphi \in C^{\ell+1}([0, 1])$, then

$$0 = \left[p_{\ell+1}(t)\varphi^{(\ell)}(t) \right]_0^1 = \int_0^1 \left[p_{\ell+1}(t)\varphi^{(\ell)}(t) \right]' dt.$$

The derivative of $p_{\ell+1}(t)$ is $\alpha_\ell - p_\ell(t)$ so that

$$\left[p_{\ell+1}(t)\varphi^{(\ell)}(t) \right]' = p_{\ell+1}(t)\varphi^{(\ell+1)}(t) + (\alpha_\ell - p_\ell(t))\varphi^{(\ell)}(t).$$

from which we deduce by an integration that

$$\int_0^1 p_{\ell+1}(t)\varphi^{(\ell+1)}(t) dt + \alpha_\ell \int_0^1 \varphi^{(\ell)}(t) dt = \int_0^1 p_\ell(t)\varphi^{(\ell)}(t) dt.$$

With this last equation, (9.12) goes from step ℓ to step $\ell + 1$.

3. $p_1(1-t) = 1/2 - (1-t) = t - 1/2 = (-1)^1 p_1(t)$.

Then, let $q_{i+1}(t) = p_{i+1}(1-t) - (-1)^{i+1} p_{i+1}(t)$. If $p_i(1-t) = (-1)^i p_i(t)$, we deduce that

$$q'_{i+1}(t) = -(\alpha_i + p_i(1-t)) - (-1)^{i+1}(\alpha_i - p_i(t)) = \lambda_{i+1} \in \mathbb{R} \quad (9.18)$$

depending on α_i and i , so that $q_{i+1}(t) = \lambda_{i+1}t + \mu_{i+1}$. Now $p_{i+1}(0) = p_{i+1}(1) = 0$, thus $q_{i+1}(0) = q_{i+1}(1) = 0$. Therefore $\lambda_{i+1} = \mu_{i+1} = 0$ and q_{i+1} is null. Finally $p_{i+1}(1-t) = (-1)^{i+1}p_{i+1}(t)$.

If we go back to (9.18) with $i = 2j+1$, we obtain

$$0 = \lambda_{2j+2} = \alpha_{2j+1}(-1 - (-1)^{2j+2}) = -2\alpha_{2j+1}. \text{ Thus } \alpha_{2j+1} = 0.$$

4. For $j \in \mathbb{N}$, let $\psi \in C^{2\ell}([j, j+1])$. For $t \in [0, 1]$, we define $\varphi(t) = \psi(t+j)$. Then $\varphi \in C^{2\ell}([0, 1])$ and (9.12) can be transformed into

$$\begin{aligned} \int_j^{j+1} \psi(t) dt &= \frac{1}{2}(\psi(j) + \psi(j+1)) + \sum_{i=1}^{\ell-1} \alpha_{2i} \int_j^{j+1} \psi^{(2i)}(t) dt \\ &\quad + \int_j^{j+1} \tilde{p}_{2\ell}(t) \psi^{(2\ell)}(t) dt. \end{aligned} \tag{9.19}$$

where 2ℓ is used instead of ℓ and also the α_{2i+1} vanish.

Summing (9.19) from $j = 1$ to $k-1$, we deduce (9.13) for $\psi \in C^{2\ell}([0, k])$.

5. For $f(x) \in C^{2\ell}([a, b])$, let $\psi(t) = f(a+th)$. The derivatives satisfy $\frac{d^i \psi}{du^i}(t) = h^i \frac{d^i f}{dx^i}(a+th)$, so that (9.13) is transformed into the Euler-Maclaurin formula, (9.14).
6. Let $f \in C^{2\ell}(\mathbb{R})$ be a $(b-a)$ -periodic function, then

$$\int_a^b f^{(2i)}(x) dx = \left[f^{(2i-1)}(x) \right]_a^b = f^{(2i-1)}(b) - f^{(2i-1)}(a) = 0.$$

In (9.14), it remains $\int_a^b f(x) dx = J(f, n) + h^{2\ell} \int_a^b \tilde{p}_{2\ell} \left(\frac{x-a}{h} \right) f^{(2\ell)}(x) dx$, where $h = (b-a)/n$.

Let $M_{2\ell} := \max_{t \in [a, b]} |f^{(2\ell)}(t)|$. Using the 1-periodicity of $\tilde{p}_{2\ell}(t)$,

$$\begin{aligned} \left| \int_a^b \tilde{p}_{2\ell} \left(\frac{x-a}{h} \right) f^{(2\ell)}(x) dx \right| &\leq M_{2\ell} \int_a^b \left| \tilde{p}_{2\ell} \left(\frac{x-a}{h} \right) \right| dx \\ &\leq M_{2\ell} h \int_0^n |\tilde{p}_{2\ell}(t)| dt \\ &= M_{2\ell} hn \int_0^1 |p_{2\ell}(t)| dt \\ &= M_{2\ell}(b-a) \int_0^1 |p_{2\ell}(t)| dt. \end{aligned}$$

We conclude that $\left| \int_a^b f(x) dx - J(f, n) \right| \leq C_\ell h^{2\ell}$ where $h = (b-a)/n$ and $C_\ell = M_{2\ell}(b-a) \int_0^1 |p_{2\ell}(t)| dt$.

7. $f_1 \in C^4(\mathbb{R})$ and $\int_0^1 f_1(x) dt = 1/30$. The numerical computation gives $\left| \int_0^1 f_1(x) dx - J(f, n) \right| \simeq C_4 h^4$.

For the second example, $f_2 \in C^\infty(\mathbb{R})$ and $\int_0^\pi f_2(x) dx = \pi/2 = J(f, n)$ up to the precision of the computation.

Solution of Exercise 9.16

1. We prove

$$(\mathcal{P}_k) : \int_a^b f(t) dt - t_k(f, n) = q_{2k, 2k} h^k + \dots + q_{2k, 2\ell} h^{2\ell} + o(h^{2\ell})$$

for $k \leq \ell$ by induction. For $k = 1$

$$\int_a^b f(t) dt - J(f, n) = q_{2,2} h^2 + q_{2,4} h^4 + \dots + q_{2,2\ell} h^{2\ell} + o(h^{2\ell}),$$

which is

$$\int_a^b f(t) dt - t_1(f, n) = q_{2,2} h^2 + q_{2,4} h^4 + \dots + q_{2,2\ell} h^{2\ell} + o(h^{2\ell}). \quad (9.20)$$

Then with n replaced by $2n$ so that h is replaced by $h/2$, we also have

$$\int_a^b f(t) dt - t_1(f, 2n) = q_{2,2} (h/2)^2 + q_{2,4} (h/2)^4 + \dots + q_{2,2\ell} (h/2)^{2\ell} + o(h^{2\ell}) \quad (9.21)$$

since $o((h/2)^{2\ell}) = o(h^{2\ell})$.

From the two equalities, (9.20) and (9.21), we deduce that

$$\begin{aligned} & 4 \left(\int_a^b f(t) dt - t_1(f, 2n) \right) - \left(\int_a^b f(t) dt - t_1(f, n) \right) \\ &= (4-1) \int_a^b f(t) dt - (4t_1(f, 2n) - t_1(f, n)) \\ &= q_{2,2} (4(h/2)^2 - h^2) + q_{2,4} ((h/2)^4 - h^4) \\ &\quad + \dots + q_{2,2\ell} ((h/2)^{2\ell} - h^{2\ell}) + o(h^{2\ell}). \end{aligned}$$

When dividing the last equality by $(4-1)$, we deduce that

$$\int_a^b f(t) dt - t_2(f, n) = q_{4,4} h^4 + \dots + q_{4,2\ell} h^{2\ell} + o(h^{2\ell}),$$

i.e. (\mathcal{P}_2) .

We deduce (\mathcal{P}_{k+1}) from (\mathcal{P}_k) by a similar method using (\mathcal{P}_k) at n and $2n$ then $4^k(h/2)^{2k} - h^{2k} = 0$.

2. romb1.m

```
p=5;
for j=0:p
    T(1,j+1)=trap(0,1,'f',2^j);
end
```

romb2.m=romb1.m+

```
for i=1:p
    mult=4^i;
    divis=mult-1;
    for j=1:p-i+1
        T(i+1,j)=(mult*T(i,j+1)-T(i,j))/divis;
    end;
end;
```

3. romb3.m

```
p=5;
for j=0:p
    T(1,j+1)=trap(0,1,'f',2^j);
end

for i=1:p
    mult=4^i;
    divis=mult-1;
    for j=1:p-i+1
        T(i+1,j)=(mult*T(i,j+1)-T(i,j))/divis;
    end;
end;
exvalue=exp(1)-1;
for i=1:p+1
    for j=1:p-i+2
        arrayerr(i,j)=abs(T(i,j)-exvalue);
    end;
end;

for i=1:p+1
    j=1:p-i+2;
    plot(log(2)*j,log(arrayerr(i,j)),'o--')
    a(i,:)=polyfit(log(2)*j,log(arrayerr(i,j)),1);
    hold on
end;
hold off
xlabel('j log 2','FontSize',14)
ylabel('log(error)','FontSize',14)
disp('Slopes of the regression lines:')
a(:,1)
```

9.6.5 A Global Quadrature

Solution of Exercise 9.17

1. f.m

```
function y=f(x)
y=exp(x);
```

2. approxinteg.m

```
function J=approxinteg(filef,n,a,b)
% global quadrature
h=(b-a)/n;
x=a+h/2:h:b-h/2;
ya=feval(filef,a);
yb=feval(filef,b);
y=feval(filef,x);
J=h*(1/9*(ya+yb)+7/8*(y(1)+y(n))...
+73/72*(y(2)+y(n-1))+sum(y(3:n-2)));
end
```

3. approxintegerr.m

```
arrn=[100:100:800]
a=-1;b=1;
filef='f';
iex=exp(b)-exp(a);
for i=1:length(arrn)
    J=approxinteg(filef,arrn(i),a,b);
    arrerr(i)=abs(J-iex);
end
plot(log(arrn),log(arrerr),'o—');
a=polyfit(log(arrn),log(arrerr),1);
title(['Slope of the regression line: ',...
    num2str(a(1))], 'FontSize',14)
xlabel('log(n)', 'FontSize',14)
ylabel('log(err)', 'FontSize',14)
```

See also Fig. 9.7.

4. approxintegerr2.m

```
arrn=[80:90]
a=-1;b=1;
filef='f';
iex=exp(b)-exp(a);
for i=1:length(arrn)
    J1=approxinteg(filef,arrn(i),a,b);
    J2=approxinteg(filef,2*arrn(i),a,b);
    J=(16*J2-J1)/15;
    arrerr(i)=abs(J-iex);
end
plot(log(arrn),log(arrerr),'o—');
```

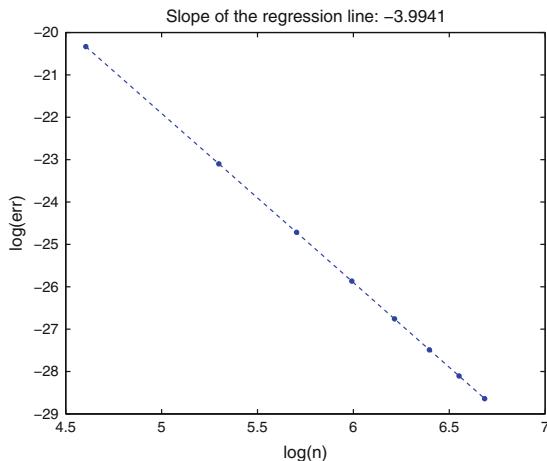


Fig. 9.7 The degree of precision of the error is 4

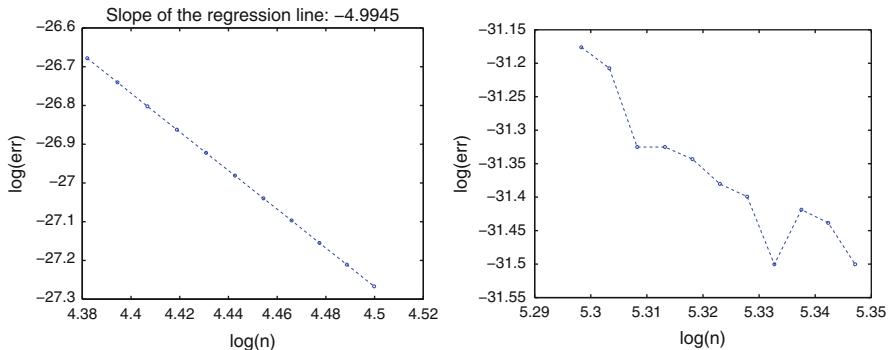


Fig. 9.8 The degree of precision of the error is 5

```
a=polyfit(log(arrn),log(arerr),1);
title(['Slope of the regression line: '...
,num2str(a(1))], 'FontSize',14)
xlabel('log(n)', 'FontSize',14)
ylabel('log(err)', 'FontSize',14)
```

See also Fig. 9.8. The degree of precision of the error is 5 but we reach the limit of MATLAB for large values of n .

Solution of Exercise 9.18

1. The construction is similar to the one of Exercise 9.14. We apply (9.16) at x_i and the global quadrature:

$$\begin{aligned}
u(x_i) &= \int_a^b K(x_i, t)u(t)dt + f(x_i) \\
&\simeq h \left\{ \frac{1}{9} [K(x_i, x_1)u(x_1) + K(x_i, x_{n+2})u(x_{n+1})] \right. \\
&\quad + \frac{7}{8} [K(x_i, x_2)u(x_2) + K(x_i, x_{n+1})u(x_{n+1})] \\
&\quad + \frac{73}{72} [K(x_i, x_3)u(x_3) + K(x_i, x_n)u(x_n)] + \sum_{j=4}^{n-1} K(x_i, x_j)u(x_j) \Big\} \\
&\quad + f_1(x_i)
\end{aligned}$$

from which we deduce

$$\begin{aligned}
v_i &= h \left\{ \frac{1}{9} [K(x_i, x_1)v_1 + K(x_i, x_{n+2})v_{n+2}] \right. \\
&\quad + \frac{7}{8} [K(x_i, x_2)v_2 + K(x_i, x_{n+1})v_{n+1}] \\
&\quad + \frac{73}{72} [K(x_i, x_3)v_3 + K(x_i, x_n)v_n] + \sum_{j=4}^{n-1} K(x_i, x_j)v_j \Big\} + f_1(x_i).
\end{aligned}$$

When collecting the previous equations for $i = 1, \dots, n+2$, we get

$$(\mathbf{I} - h\mathbf{A}_n)\mathbf{V} = \mathbf{F}.$$

2. `inteqnew1.m`

```

n=3;
a=-1;b=1;
h=(b-a)/n;
x=[a , a+h/2:h:b-h/2 , b];
X=x*ones(1 , n+2)

```

3. `K.m`

```

function z=K(x,y);
z=(x-y).*(x-y);

```

4. `inteqnew1.m=inteqnew1.m+`

```

B=K(X,X');
An=B;
An(:,1)=1/9*An(:,1);
An(:,n+2)=1/9*An(:,n+2);
An(:,2)=7/8*An(:,2);
An(:,n+1)=7/8*An(:,n+1);
An(:,3)=73/72*An(:,3);
An(:,n)=73/72*An(:,n);

```

5. `inteqnew3.m=inteqnew2.m+`

```

F=f1(x);
V=(eye(n+2)-h*An)\F;
U=u(x);

```

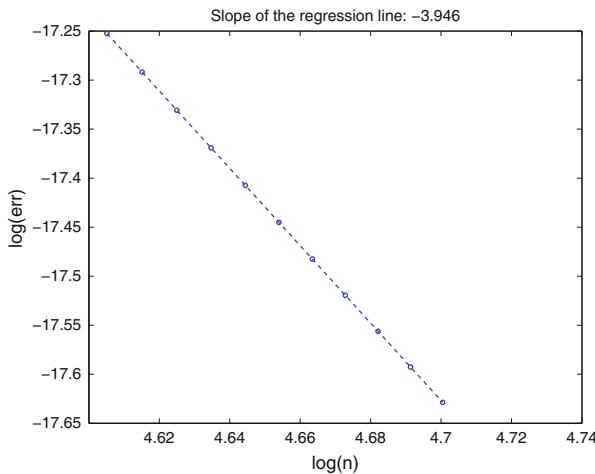


Fig. 9.9 The degree of precision is 4

```
err=norm(V-U, inf)
t=a:(b-a)/500:b;
plot(x,U, 'ob', t,u(t), 'b', x,V, '--r');
```

6. inteqnew4.m

```
arrn=100:110;
a=-1;b=1;
for i=1:length(arrn)
    n=arrn(i);
    h=(b-a)/n;
    x=[a, a+h/2:h:b-h/2,b]';
    X=x*ones(1,n+2);
    An=K(X,X');
    An(:,1)=1/9*An(:,1);
    An(:,n+2)=1/9*An(:,n+2);
    An(:,2)=7/8*An(:,2);
    An(:,n+1)=7/8*An(:,n+1);
    An(:,3)=73/72*An(:,3);
    An(:,n)=73/72*An(:,n);
    F=f1(x);
    V=(eye(n+2)-h*An)\F;
    U=u(x);
    arrerr(i)=norm(V-U, inf);
end
plot(log(arrn),log(arrerr), 'o--');
a=polyfit(log(arrn),log(arrerr),1);
title([' Slope of the regression line: ', ...
    num2str(a(1))], 'FontSize', 14)
xlabel('log(n)', 'FontSize', 14)
ylabel('log(err)', 'FontSize', 14)
```

See Fig. 9.9

Chapter 10

Linear Least Squares Methods

The method of least squares is the standard method for finding an approximate solution of a linear system with more equations than unknowns. Such systems occur in data fitting where one determines the fit by minimizing the sum of squares of the difference between the observed and the fitted values. The linear least squares problem also occurs in statistical regression analysis, in signal processing, and in many other applications.

Theorem 10.1 shows that the least squares solution can be found by solving a square linear system known as the **Normal equations**. The proof and necessary tools are given in Sect. 10.1, see Exercise 10.7 and Exercises 10.1, 10.4 and 10.5, while Exercises 10.2, 10.3, 10.6 and 10.8 are given to complete the understanding of the reader by examples. The question of the uniqueness of the solution is studied in Exercise 10.9.

Some readers might accept Theorem 10.1 without proof and prefer to go directly to Sect. 10.2. There we give several curve fitting examples leading to linear least squares problems. We start with fitting polynomials of degree 1 or 2 to data in Exercise 10.10. A least square approximation using conics in the plane is developed in Exercise 10.11. In Sect. 10.3 we use least squares to recover a periodic signal from a noisy one in Exercise 10.12, using trigonometric functions. This is followed by the Savistky-Golay Filter in Exercise 10.13, that uses a local least square method based on polynomials. We conclude the chapter with Exercise 10.14, where a differential equation is solved approximately using a least squares method.

Another example of least squares problem is given in Chap. 4.

Review: Suppose $m, n \in \mathbb{N}$, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. The rectangular system $Ax = b$ can have either no solution, one solution or an infinite number of solutions. Let $\|e\|_2 := \sqrt{e_1^2 + \cdots + e_m^2}$ be the Euclidean norm of a vector $e \in \mathbb{R}^m$. We can find an approximate solution by minimizing $J(\alpha) := \|A\alpha - b\|_2^2$. This is called the **least squares problem** and any $\alpha \in \mathbb{R}^n$ minimizing $J(\alpha)$ is called a **least squares solution**. Using orthogonality, see Sect. 10.1 we will show

Theorem 10.1. *The following holds for $m, n \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\mathbf{b} \in \mathbb{R}^m$.*

1. *The least squares problem always has a solution.*
2. *The least squares solution is unique if and only if $\text{rank}(\mathbf{A}) = n$.*
3. *α is a least squares solution if and only if $\mathbf{A}^T \mathbf{A}\alpha = \mathbf{A}^T \mathbf{b}$.*
4. *The matrix $\mathbf{B} := \mathbf{A}^T \mathbf{A}$ is symmetric and positive definite and hence nonsingular if and only if $\text{rank}(\mathbf{A}) = n$.*

Recall that a matrix $\mathbf{B} \in \mathbb{R}^{n \times n}$ is symmetric if $\mathbf{B}^T = \mathbf{B}$ and positive definite if $\alpha^T \mathbf{B} \alpha > 0$ for all nonzero $\alpha \in \mathbb{R}^n$. The linear system $\mathbf{A}^T \mathbf{A}\alpha = \mathbf{A}^T \mathbf{b}$ is known as the **normal equations**. Orthogonality plays an important role and is a key argument in least squares problems and is one of the main reasons for using the Euclidean norm to find an approximate solution of a rectangular system $\mathbf{Ax} = \mathbf{b}$.

▲

10.1 Orthogonal Projections and Orthogonal Sums

► **Review:** In this section we first assume that \max is a finite dimensional subspace of a vector space \mathcal{E} where an inner product denoted $\langle \cdot, \cdot \rangle$ is defined. Recall that it satisfies bilinearity, symmetry and $\langle \mathbf{x}, \mathbf{x} \rangle > 0$ for any $\mathbf{x} \neq \mathbf{0}$. ▲ Two examples are

Example 10.1. $\mathcal{E} = \mathbb{R}^m$, $\max := \text{span}(\mathbf{A})$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^T \mathbf{y}$.

Example 10.2. $\mathcal{E} = C[-\pi, \pi]$, $\max_n := \text{span}(\{1, \cos x, \sin x, \dots, \cos nx, \sin nx\})$, and $\langle f, g \rangle = \int_{-\pi}^{\pi} f(x)g(x)dx$.

Exercise 10.1. Orthogonal projection

1. Suppose $\mathbf{b} \in \mathcal{E}$ and that $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ is an orthogonal basis for \max , i.e., $\langle \mathbf{v}_i, \mathbf{v}_j \rangle = 0$ if $i \neq j$. If

$$\mathbf{b}_{\max} := \sum_{i=1}^k \frac{\langle \mathbf{b}, \mathbf{v}_i \rangle}{\langle \mathbf{v}_i, \mathbf{v}_i \rangle} \mathbf{v}_i \quad (10.1)$$

prove that

$$\langle \mathbf{b} - \mathbf{b}_{\max}, \mathbf{f} \rangle = 0, \quad \text{for all } \mathbf{f} \in \max. \quad (10.2)$$

► *Math Hint*¹ ◁

2. Show that to each $\mathbf{b} \in \mathcal{E}$ there is a unique $\mathbf{b}_{\max} \in \max$ satisfying (10.2). Thus (10.2) and (10.1) are equivalent. The vector \mathbf{b}_{\max} is called the **orthogonal projection of \mathbf{b} on \max** . See Fig. 10.1 ► *Math Hint*² ◁

¹ Show first that $\langle \mathbf{b}_{\max}, \mathbf{v}_j \rangle = \langle \mathbf{b}, \mathbf{v}_j \rangle$ for $j = 1, 2, \dots, k$.

² Show that if \mathbf{b}_{\max_1} and \mathbf{b}_{\max_2} both satisfies (10.2) then $\mathbf{b}_{\max_1} = \mathbf{b}_{\max_2}$.

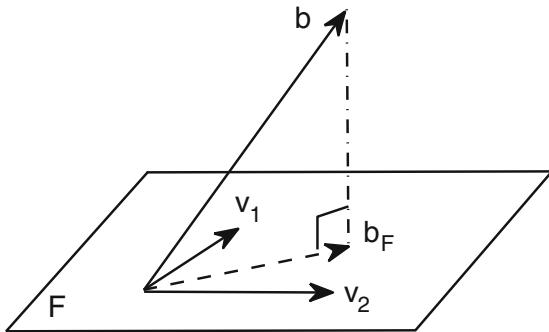


Fig. 10.1 Orthogonal Projection of b onto \max

Exercise 10.2. Column space projection

Given $A := \begin{bmatrix} 1 & 1 \\ 1 & -2 \\ 1 & 1 \end{bmatrix}$. Use (10.1) to find the projection of $b := \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$ onto the subspaces $\max := \text{span}(A)$ and $\mathcal{G} := \ker(A^T)$ of \mathbb{R}^3 . You should obtain $b = b_{\max} + b_{\mathcal{G}}$ (This is generalized in Exercise 10.5)

Exercise 10.3. (Fourier projection)

Jean-Baptiste Joseph Fourier (1768–1830) was a French mathematician and physicist best known for initiating the investigation of Fourier series and their applications to problems of heat transfer and vibrations. The Fourier transform and Fourier's Law are also named in his honor.



Consider the subspace $\max_n := \text{span}(\{1, \cos x, \sin x, \dots, \cos nx, \sin nx\})$ of $\mathcal{E} = C[-\pi, \pi]$ with inner product $\langle f, g \rangle = \int_{-\pi}^{\pi} f(x)g(x)dx$ (cf. Example 10.2).

1. Show that $(\{1, \cos x, \sin x, \dots, \cos nx, \sin nx\})$ is an orthogonal basis for \max_n .
2. Find the orthogonal projection of $f \in C[-\pi, \pi]$ onto \max_n .

Review: Recall that the sums of subspaces are defined by

$$\mathcal{H} = \max + \mathcal{G} := \{f + g : f \in \max, g \in \mathcal{G}\},$$

$$\mathcal{H} = \max \oplus \mathcal{G} \text{ if } \mathcal{H} = \max + \mathcal{G} \text{ and } \max \cap \mathcal{G} = \{\mathbf{0}\},$$

$$\mathcal{H} = \max \overset{\perp}{\oplus} \mathcal{G} := \{f + g : f \in \max, g \in \mathcal{G}\} \text{ and } \langle f, g \rangle = 0, f \in \max, g \in \mathcal{G}$$

$\mathcal{H} = \max \oplus \mathcal{G}$ is called a **direct sum**, $\mathcal{H} = \max \overset{\perp}{\oplus} \mathcal{G}$ is known as an **orthogonal sum**, and $\mathcal{H} = \max \overset{\perp}{\oplus} \mathcal{G}$ is called an **orthogonal decomposition** of \mathcal{H} .

Exercise 10.4. Orthogonal sum: $\mathcal{H} = \max \bigoplus^{\perp} \mathcal{G}$

1. Show that if $\mathbf{h} \in \max \cap \mathcal{G}$ then $\mathbf{h} = \mathbf{0}$, i.e., an orthogonal sum is also a direct sum.
2. Show that $\mathbf{h} \in \max \bigoplus^{\perp} \mathcal{G}$ can be decomposed uniquely as $\mathbf{h} = \mathbf{f} + \mathbf{g}$, where $\mathbf{f} \in \max$ and $\mathbf{g} \in \mathcal{G}$. ▷ Math Hint³ ◁
3. If $\mathbf{h} = \mathbf{f} + \mathbf{g}$, where $\mathbf{f} \in \max$ and $\mathbf{g} \in \mathcal{G}$ then

$$\|\mathbf{f} + \mathbf{g}\|_2^2 = \|\mathbf{f}\|_2^2 + \|\mathbf{g}\|_2^2 \quad (\text{Pythagoras}),$$

where the norm is computed from the inner product $\|\mathbf{f}\| := \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle}$.

Pythagoras of Samos (about 569–about 475 BC) was an Ionian Greek philosopher, mathematician, and founder of the religious movement called Pythagoreanism. He made influential contributions to philosophy and religious teaching in the late 6th century BC. He is often revered as a great mathematician, mystic and scientist, but he is best known for the Pythagorean theorem which bears his name.


Exercise 10.5. Subspaces of a matrix

Consider for any $m, n \in \mathbb{N}$ and any $\mathbf{A} \in \mathbb{R}^{m \times n}$ the subspaces of \mathbb{R}^m $\text{span}(\mathbf{A}) := \{\mathbf{y} = \mathbf{Ax} : \mathbf{x} \in \mathbb{R}^n\}$ and $\ker(\mathbf{A}^T) := \{\mathbf{y} \in \mathbb{R}^m : \mathbf{A}^T \mathbf{y} = \mathbf{0}\}$ with the usual inner product $\langle \mathbf{x}, \mathbf{y} \rangle := \mathbf{x}^T \mathbf{y}$.

1. Show that the sum of $\text{span}(\mathbf{A})$ and $\ker(\mathbf{A}^T)$ is an orthogonal sum. ▷ Math Hint⁴ ◁
2. Show that $\mathbb{R}^m = \text{span}(\mathbf{A}) \bigoplus^{\perp} \ker(\mathbf{A}^T)$. ▷ Math Hint⁵ ◁

Exercise 10.6. Let a subspace of \mathbb{R}^2 be given by $\mathcal{G} := \{(x, y) \in \mathbb{R}^2 : 3x - y = 0\}$.

1. Find a matrix \mathbf{A} such that $\mathcal{G} = \ker(\mathbf{A}^T)$.
2. Find the orthogonal projections \mathbf{b}_1 and \mathbf{b}_2 onto $\text{span}(\mathbf{A})$ and $\ker(\mathbf{A}^T)$ of $\mathbf{b} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$. ▷ Math Hint⁶ ◁

Exercise 10.7. In this problem we lead you through a proof of Theorem 10.1. It is recommended to first work through Exercises 10.1, 10.4 and 10.5. Suppose $m, n \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{b} \in \mathbb{R}^m$ are given. We recall that $\mathbb{R}^m = \text{span}(\mathbf{A}) \bigoplus^{\perp} \ker(\mathbf{A}^T)$ (cf. Exercise 10.5). Let $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where \mathbf{b}_1 and \mathbf{b}_2 are the orthogonal projections onto the subspaces $\max := \text{span}(\mathbf{A})$ and $\mathcal{G} := \ker(\mathbf{A}^T)$.

³ Suppose $\mathbf{h} = \mathbf{f}_1 + \mathbf{g}_1 = \mathbf{f}_2 + \mathbf{g}_2$. Show that $\mathbf{f}_1 - \mathbf{f}_2 \in \max \cap \mathcal{G}$.

⁴ If $\mathbf{f} \in \text{span}(\mathbf{A})$ then $\mathbf{f} = \mathbf{Ax}$ for some \mathbf{x} .

⁵ Let $\mathbf{b} \in \mathbb{R}^m$. Define \mathbf{b}_1 as the orthogonal projection of \mathbf{b} onto $\text{span}(\mathbf{A})$ and use (10.2).

⁶ Use (10.1) to find \mathbf{b}_1 .

1. Show that $(\mathbf{A}\mathbf{x} - \mathbf{b}_1)^T \mathbf{b}_2 = 0$ for all $\mathbf{x} \in \mathbb{R}^n$.
2. Write $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2 = \|(\mathbf{A}\mathbf{x} - \mathbf{b}_1) + \mathbf{b}_2\|_2^2$ and use Pythagoras (cf. Question 3 of Exercise 10.4) to show that any $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{A}\mathbf{x} = \mathbf{b}_1$ is a least squares solution. Thus existence follows.
3. Show that the system $\mathbf{A}\mathbf{x} = \mathbf{b}_1$ has a unique solution \mathbf{x} if and only if $\text{rank}(\mathbf{A}) = n$. This shows uniqueness.
4. Prove that if $\mathbf{A}\mathbf{x} = \mathbf{b}_1$ then $\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{0}$. Thus a least squares solution satisfies the normal equations.
5. Show the converse, if $\mathbf{A}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) = \mathbf{0}$ then $\mathbf{A}\mathbf{x} = \mathbf{b}_1$. \triangleright Math Hint⁷ \triangleleft
6. Show that the matrix $\mathbf{B} := \mathbf{A}^T \mathbf{A}$ is symmetric and positive definite, and hence nonsingular, if and only if $\text{rank}(\mathbf{A}) = n$.
7. Suppose $m, n \in \mathbb{N}$, $\mathbf{A} \in \mathbb{R}^{m \times n}$ has rank n , and that $\mathbf{b} \in \mathbb{R}^m$. The projection \mathbf{b}_1 of \mathbf{b} onto $\text{span}(\mathbf{A})$ can be written

$$\mathbf{b}_1 = \mathbf{A}\mathbf{A}^\dagger \mathbf{b}, \quad \mathbf{A}^\dagger := (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T.$$

\triangleright Math Hint⁸ \triangleleft

8. Suppose \mathbf{A} is square and nonsingular. Show that $\mathbf{A}^{-1} = \mathbf{A}^\dagger$. The matrix \mathbf{A}^\dagger is called the **generalized inverse** of \mathbf{A} .

Exercise 10.8. Let \max be a subspace of an inner product space \mathcal{E} . Let $\mathcal{G} = \{\mathbf{g} \in \mathcal{E} : \langle \mathbf{f}, \mathbf{g} \rangle = 0, \mathbf{f} \in \max\}$. Prove that \mathcal{G} is a subspace of \mathcal{E} and if \max is finitely dimensional then $\mathcal{E} = \max \overset{\perp}{\oplus} \mathcal{G}$.

10.2 Curve Fitting by Least Squares

Review: Given data $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ where the components x_1, \dots, x_m of \mathbf{x} belong to an interval $I \subset \mathbb{R}$ and let $\{\varphi_1, \varphi_2, \dots, \varphi_n\}$ be n functions defined on I with $m \geq n$. We are looking for a function $\phi_\alpha = \sum_{j=1}^n \alpha_j \varphi_j$ which approximate the data

i.e. $\phi_\alpha(x_i) \approx y_i$. This curve fitting problem can be defined from the overdetermined linear system $\mathbf{A}\alpha = \mathbf{b}$, where $\mathbf{b} := \mathbf{y}$ and

$$\mathbf{A}\alpha = \begin{bmatrix} \phi_\alpha(x_1) \\ \vdots \\ \phi_\alpha(x_m) \end{bmatrix} \in \mathbb{R}^m, \quad \mathbf{A} = \begin{bmatrix} \varphi_1(x_1) & \cdots & \varphi_n(x_1) \\ \vdots & & \vdots \\ \varphi_1(x_m) & \cdots & \varphi_n(x_m) \end{bmatrix} \in \mathbb{R}^{m \times n}. \quad (10.3)$$

¶

⁷ Use the fact that $\max \cap \mathcal{G} = \{\mathbf{0}\}$.

⁸ The least squares solution is $\mathbf{x} = \mathbf{A}^\dagger \mathbf{b}$.

Then, we find $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_n]^T \in \mathbb{R}^n$ as a solution of the corresponding least squares problem given by

$$J(\alpha) := \|\mathbf{A}\alpha - \mathbf{b}\|_2^2 = \sum_{i=1}^m (\phi_\alpha(x_i) - y_i)^2. \quad (10.4)$$

Exercise 10.9. Conditions for uniqueness

- As shown in Exercise 10.7 the least squares problem always has a solution. Show that if

$$\phi_\alpha(x_i) = 0, \quad i = 1, \dots, m \Rightarrow \alpha = \mathbf{0},$$

then $\text{rank}(\mathbf{A}) = n$ and the least squares problem has a unique solution.

- Consider the least squares problem with $\varphi_j(x) = x^{j-1}$ for $j = 1, \dots, n$. If x has at least n distinct components then the least squares problem has a unique solution and $\mathbf{A}^T \mathbf{A}$ is symmetric and positive definite.

Exercise 10.10. Linear and Parabolic Regression:

In this first example, we use the data

i	1	2	3	4	5
x	-2	-1	0	1	2
y	3/2	-1	1	5	17/2

and the functions $\varphi_j : I = [-2, 2] \rightarrow \mathbb{R}$ given by $\varphi_j(x) = x^{j-1}$, $j = 1, 2, \dots$. We first want to fit the data with a straight line and then with a parabola i.e., minimizing $\sum_{i=1}^5 (y_i - \alpha_1 - \alpha_2 x_i)^2$ and then $\sum_{i=1}^5 (y_i - \alpha_1 - \alpha_2 x_i - \alpha_3 x_i^2)^2$.

- Find the normal equations and the least squares solution ϕ_α by hand calculation for a straight line and a parabola. \triangleright *Math Hint*⁹ \triangleleft
- Programs:** Then use the MATLAB function `polyfit` to find the coefficients α . Make graphs for the straight line and the parabola showing the points as well as the curves using MATLAB. See Fig. 10.2. \triangleleft *MATLAB Hint*¹⁰ \triangleright

Exercise 10.11. Least Squares in the Plane

In this problem we think of x and y as points $\mathbf{p}_i = (x_i, y_i)$, $i = 1, \dots, m$, in the plane \mathbb{R}^2 , and use the quadratic functions $\varphi_j : \mathbb{R}^2 \rightarrow \mathbb{R}$ given for $j = 1, \dots, 6$ by $\varphi_1(x, y) = x^2, \varphi_2(x, y) = y^2, \varphi_3(x, y) = xy, \varphi_4(x, y) = x, \varphi_5(x, y) = y, \varphi_6(x, y) = 1$. Let $\phi_\alpha = \sum_{j=1}^5 \alpha_j \varphi_j - \varphi_6$. We want to fit a conic

⁹ Use (10.3).

¹⁰ Note the ordering of the coefficient of a polynomial in MATLAB. For example $p(x) = \alpha_2 + \alpha_1 x$ in the linear case.

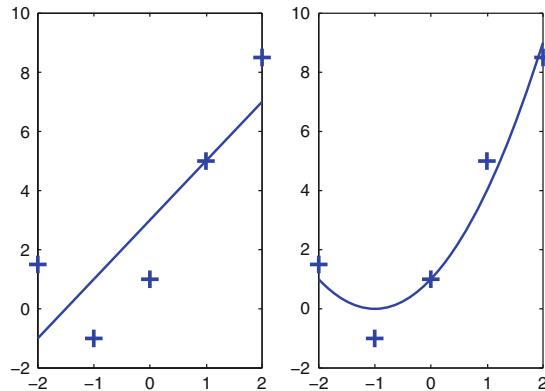


Fig. 10.2 Linear and quadratic regression

section to p_1, \dots, p_m by minimizing

$$J(\alpha) := \sum_{i=1}^m \phi_\alpha(x_i, y_i)^2.$$

1. Show that $J(\alpha) = \|A\alpha - b\|_2^2$ for some $A \in \mathbb{R}^{m \times 5}$ and $b = [1, \dots, 1]^T \in \mathbb{R}^m$.
2. We fix $m = 8$ and the points

$$\begin{aligned} p_1 &= (0, 1), & p_2 &= (2 - \sqrt{3}, \frac{1}{2}), & p_3 &= (2, 0), & p_4 &= (2 + \sqrt{3}, \frac{1}{2}), \\ p_5 &= (4, 1), & p_6 &= (2 + \sqrt{3}, \frac{3}{2}), & p_7 &= (2, 2), & p_8 &= (2 - \sqrt{3}, \frac{3}{2}). \end{aligned}$$

Find, using hand calculation, the corresponding matrix A .

3. **Programs:** Write a function $M=mat(x, y)$ which computes the values of $M_{i,j} = \varphi_j(x_i, y_i)$ for $i = 1, \dots, m$, and $j = 1, \dots, 6$. Test with $x = [1, -1, 2]', y = [0, 3, 4]'$. \triangleleft MATLAB Hint¹¹

```
>> M=mat([1, -1, 2]', [0, 3, 4]')
M =
 1      0      0      1      0      1
 1      9     -3     -1      3      1
 4     16      8      2      4      1
```

4. Write a program `conic1.m` which computes the matrix A and the vector b from a given pair (x, y) . Test: $x = [1, -1, 2]', y = [0, 3, 4]'$.

¹¹ To compute x^2 use the instruction `x.*x`.

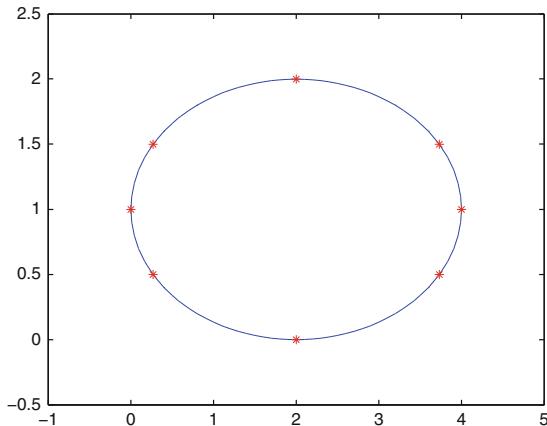


Fig. 10.3 Ellipse

```
>> conic1
A =
    1      0      0      1      0
    1      9     -3     -1      3
    4     16      8      2      4

b =
    1
    1
    1
```

5. Write a function `phi.m` which to given x, y, α computes the array $\phi_\alpha(x, y) = \sum_{j=1}^6 \alpha_j \varphi_j(x, y)$. Test:

```
>> X=magic(3);Y=X;c=ones(6,1);Z=phi(X,Y,c)
Z =
    209      6     121
    34     86     162
    57    262      17
```

6. Extend the program `conic1.m` into `conic2.m` with the computation of the coefficients α . Add the construction of a graph with the given points x, y and the conic $\phi_\alpha(x, y) = 0$. To compute the curve you can use the instructions
`[Xg, Yg] = meshgrid([xmin:deltax:xmax], [ymin:deltay:ymax])`,
`contour(Xg, Yg, Zg, [0 0])`
where Zg is computed using the function `phi`. Test with x, y given in Question 2. See Fig. 10.3.

```
>> conic2
c =
    0.2500
    1.0000
```

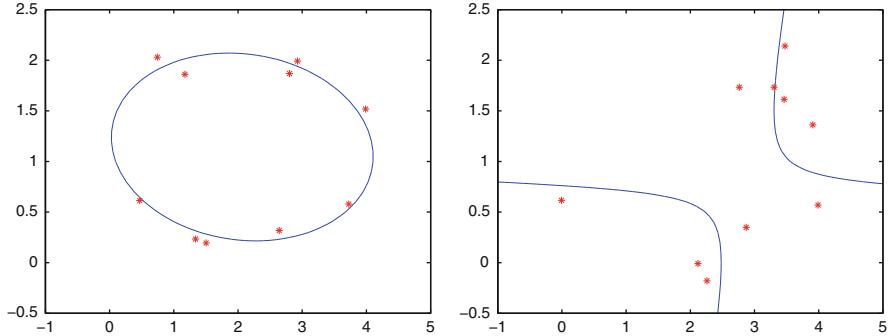


Fig. 10.4 Ellipse and Hyperbola

-0.0000
-1.0000
-2.0000
1.0000

7. Try with $m = 50$ and random points sampled from the ellipse $x(t) = 2 + 2 \cos(t)$, $y(t) = 1 + \sin(t)$.
8. Run the program many times with the noisy values

```
theta=2*pi*rand(m,1); x=1.8+2*cos(theta)+0.6*rand(m,1);
x=0.8+sin(theta)+0.6*rand(m,1);
```

with $m = 10$. See Fig. 10.4.

10.3 Periodic Signal

Exercise 10.12. We want to recover a periodic signal $y : [-1, 1] \rightarrow \mathbb{R}$ from sampled values y_{n_1}, \dots, y_{n_m} at points $-1 = x_1 < \dots < x_m = 1$. Thus, y_{n_k} is an approximation to $y(x_k)$ containing unknown noise. We assume first that the signal is odd, i.e., $y(-t) = -y(t)$ for $t \in [-1, 1]$. We want to recover an approximation $y_s : [-1, 1] \rightarrow \mathbb{R}$ to y using a least squares method.

For any positive integer n , this is done by the minimization of

$$E(p) = \sum_{i=1}^m (y_{n_i} - p(x_i))^2 \text{ where } p(x) = \sum_{j=1}^n u_j \sin(j\pi x).$$

We are looking for $\alpha = [\alpha_1, \dots, \alpha_n]^T$, a solution of the system $\mathbf{A}^T \mathbf{A} \alpha = \mathbf{A}^T \mathbf{y}_n$ where $\mathbf{A} \in \mathbb{R}^{m,n}$ is the rectangular matrix with $a_{ij} = \sin(j\pi x_i)$.

The noisy signal is obtained using the program `signal11.m` with input m and output the three vectors \mathbf{x} , \mathbf{y} , \mathbf{y}_n , where \mathbf{x} are the abscissas which are random values in the interval $[-1, 1]$ except the first and last ones (the abscissas are sorted),

$\mathbf{y} = [y_1, \dots, y_m]$ with $y_i = y(x_i)$ is the exact signal (which will be only used to check the result), and \mathbf{yn} the noisy signal.

```
function [x,y,yn]=signal1(m);
x=sort(2*rand(m,1)-1);
x(1)=-1;x(m)=1;
y=sin(6*pi*x);
yn=y+(2*rand(m,1)-1)*0.6;
```

1. Write a program `SP1.m` with input m and n , which calls `signal1(m)`, then constructs and outputs the matrix \mathbf{A} . You can start by constructing a matrix \mathbf{B} (using an outer product construction and no loops) such that $b_{ij} = x_i \times j$. Test with `SP1(6, 3)`.

```
>> SP1(6,3)
A =
-0.0000    0.0000   -0.0000
-0.8714    0.8550    0.0326
 0.4297    0.7760    0.9718
 0.6085    0.9658    0.9242
 0.6830    0.9977    0.7747
 0.0000   -0.0000    0.0000
```

©Comment: Since the abscissas are random, the constructed matrix \mathbf{A} can differ a bit from what is shown in the solution part. ☺

2. Complete the first program into `SP2.m` which solves the system $\mathbf{A}^T \mathbf{A} \boldsymbol{\alpha} = \mathbf{A}^T \mathbf{y}$. Output $\boldsymbol{\alpha}$. Test with `SP2(1000, 8)`.

```
>> SP2(1000,8)
alpha =
  0.0152
 -0.0139
  0.0056
  0.0262
  0.0072
  1.0043
  0.0138
 -0.0161
```

©Comment: The abscissas and the noise are random so that the output values will differ. In any case the sixth component is almost 1 and the others close to 0. ☺

3. Extend the second program into `SP3.m` by the computation of the smooth signal $ys_k = \sum_{j=1}^n \alpha_j \sin(j\pi x_k)$, $k = 1, \dots, m$ (without using loops), then plot four figures using `subplot`: initial signal \mathbf{y} , noisy signal \mathbf{yn} , smooth signal \mathbf{ys} and the superposition of \mathbf{y} and \mathbf{ys} . Test `SP3(1000, 8)`, see Fig. 10.5. Try another test `SP3(1000, 5)`. See Fig. 10.6. Why do we get this result? Where is the noise in the previous case?

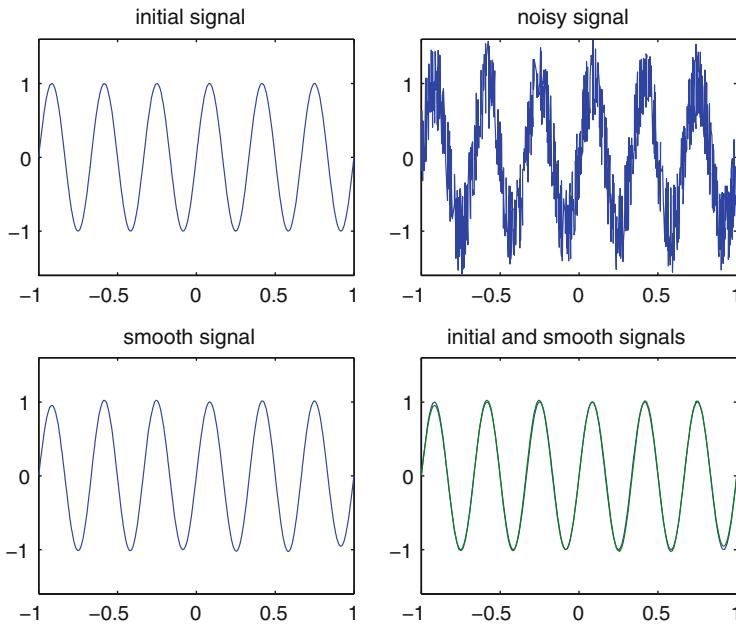


Fig. 10.5 Figures from the call `SP3(1000,8)`, see Question 3

4. In this part the signal is no longer odd but still periodic. We use the same tools with the $n := 2N + 1$ functions

$$\{1, \sin \pi t, \cos \pi t, \sin 2\pi t, \cos 2\pi t, \dots, \sin N\pi t, \cos N\pi t\}.$$

and `signal2.m`

```
function [x,y,yn]=signal2(m);
x=sort(2*rand(m,1)-1);
x(1)=-1;x(m)=1;
y=0.3+cos(10*pi*x)+sin(4*pi*x);
yn=y+(2*rand(m,1)-1)*0.6;
```

Write a new program `SP4.m` with input m and N , which calls `signal2`, then computes the smooth signal and plot the four figures as in the previous question. Test using `SP4(1000,11)`, see Fig. 10.7.

```
>> SP4(1000,11)
u =
    0.2944
    0.0104
   -0.0094
   -0.0023
   -0.0063
   -0.0097
   -0.0218
    1.0357
    0.0056
```

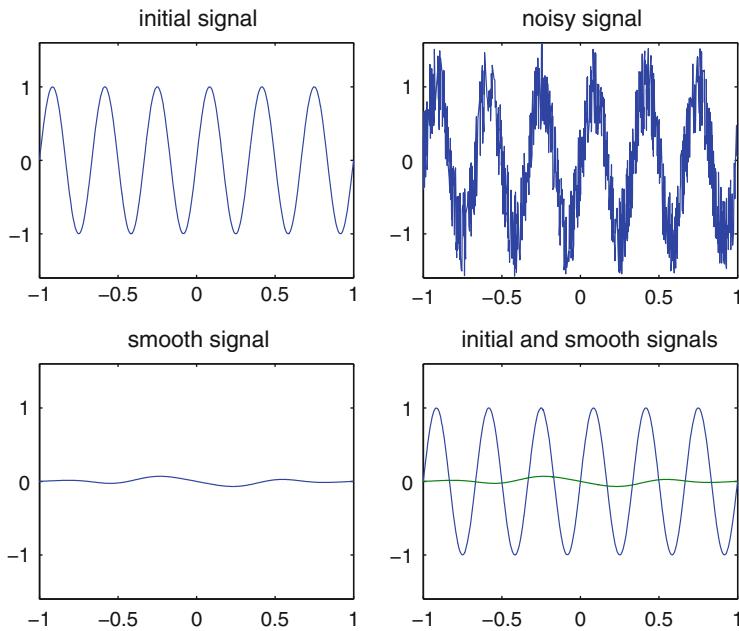


Fig. 10.6 Figures from the call `SP3(1000,5)`, see Question 3

0.0076
-0.0148
-0.0131
-0.0094
-0.0052
0.0216
0.0259
-0.0124
0.0064
0.0025
0.0143
1.0249
-0.0424
-0.0098

10.4 Savitsky-Golay Filter

Marcel J.E. Golay (1902–1989) was a Swiss-born mathematician, physicist who applied mathematics to industrial problems. In USA, he worked on many problems, including gas chromatography and optical spectroscopy. In mathematics and electronics engineering, a binary Golay code is a type of error-correcting code used in digital communications.



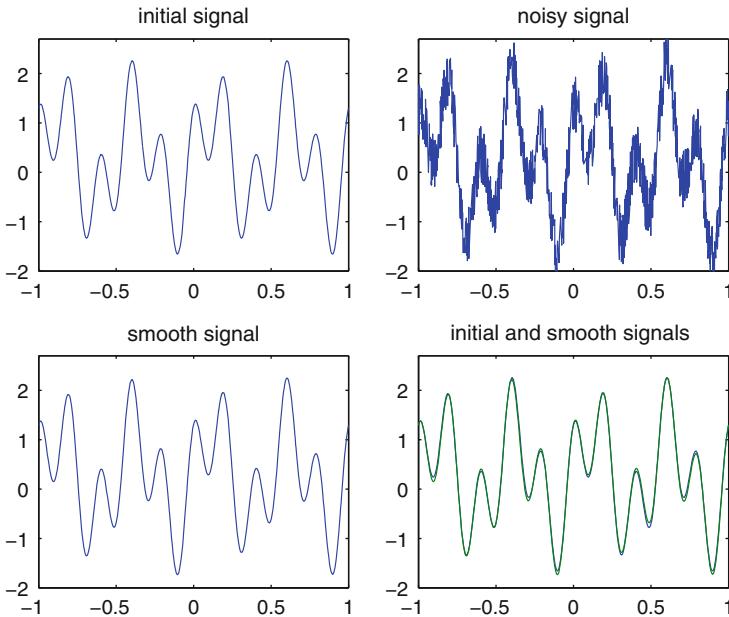


Fig. 10.7 SP4(1000,11)

Abraham Savitzky (1919–1999) was an American analytical chemist. He specialized in the digital processing of infrared spectra. While employed by Perkin-Elmer, Savitzky coauthored with Marcel J. E. Golay an oft-cited paper describing the Savitzky-Golay Smoothing Filter for digital filtering.

Exercise 10.13. The least squares method can be used for local smoothing of noisy data $(x_k, f_k) \in \mathbb{R}^2$, $k = 1, \dots, p$. We assume error free uniform abscissas, $x_{k+1} - x_k = \Delta x > 0$, $k = 1, \dots, p-1$, and want to smooth the values f_k that are supposed to be noisy. For a given index k , the idea is to substitute for f_k a value g_k computed from an average of f -values close to f_k . More precisely, let n_ℓ, n_r, n be positive integers. At a point x_k with $n_\ell + 1 \leq k \leq p - n_r$, we use the neighbours with abscissas $x_{k-n_\ell}, \dots, x_k, \dots, x_{k+n_r}$. We compute the polynomial p_k of degree less than n smoothing the $m := n_\ell + n_r + 1$ points, in the sense that p_k gives the minimum value of $\sum_{i=k-n_\ell}^{k+n_r} (p(x_i) - f_i)^2$, $p \in \mathbb{P}_{n-1}$. Then we define $g_k := p_k(x_k)$.

Comment: The number of values n_ℓ and n_r on the left and on the right of x_k is to be given by the user. This is also the case for n , where $n-1$ is the degree of the polynomial. ☺

The choice of the functions $\varphi_j(x) := \left(\frac{x-x_k}{\Delta x}\right)^{j-1}$ in \mathbb{P}_{n-1} is a key point of the method. We are looking for $p_k = \sum_{j=1}^n \alpha_j \varphi_j \in \mathbb{P}_{n-1}$ minimizing

$$J(\boldsymbol{\alpha}) := \sum_{i=k-n_\ell}^{k+n_r} (p_k(x_i) - f_i)^2.$$

This is a curve fitting by least squares problem and can be written $\|\mathbf{A}\boldsymbol{\alpha} - \mathbf{b}\|_2^2$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $a_{i,j} = \varphi_j(x_{k+i-n_\ell-1})$ and $b_i = f_{k+i-n_\ell-1}$, for $i = 1, \dots, m$ and $j = 1, \dots, n$.

1. Show that $\left\{ \varphi_j(x) := \left(\frac{x - x_k}{\Delta x} \right)^{j-1} \right\}_{j=1, \dots, n}$ is a basis of \mathbb{P}_{n-1} .
2. Determine by hand calculation \mathbf{A} using the current basis. Notice that \mathbf{A} is independent of k .
3. Show that \mathbf{A} is of rank n so that the minimization problem has a unique solution $\boldsymbol{\alpha}$ such that $\mathbf{A}^T \mathbf{A} \boldsymbol{\alpha} = \mathbf{A}^T \mathbf{b}$. *Math Hint*¹² □
4. Let $p_k = \sum_{j=1}^n \alpha_j \varphi_j$. If $g_k := p_k(x_k)$, show that $g_k = \sum_{i=-n_\ell}^{n_r} c_i f_{k+i}$, where $\mathbf{c}^T = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{e}_1$ is the first column of $\mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1}$. Note that $\mathbf{c} \in \mathbb{R}^m$ is independent of k so it only needs to be computed once. *Math Hint*¹³ □
5. **Programs:** Write the function `signal1.m` that samples

$$\begin{aligned} F(x) &= \frac{1}{1 + 100(x - 0.2)^2} + \frac{1}{1 + 500(x - 0.4)^2} \\ &\quad + \frac{1}{1 + 1000(x - 0.6)^2} + \frac{1}{1 + 10000(x - 0.8)^2} \end{aligned}$$

at p uniform points in $[0, 1]$. Test with $p = 5$, outputs \mathbf{x} and \mathbf{y} .

```
>> [x, y] = signal1(5)
x =
      0      0.2500      0.5000      0.7500      1.0000
y =
    0.2153      0.8901      0.3587      0.1291      0.0296
```

6. Add a noise to the initial signal, $\mathbf{y}_n = \mathbf{y} + \frac{10}{100} \mathbf{v}$ where \mathbf{v} is a random vector with p components between -1 and 1 . You can use the MATLAB function `rand.m` which gives random number in the interval $[0, 1]$. Save in `signal2.m`. Test with $p = 5$.

```
>> [x, y, yn] = signal2(5)
x =
      0      0.2500      0.5000      0.7500      1.0000
y =
    0.2153      0.8901      0.3587      0.1291      0.0296
```

¹² Vandermonde matrix.

¹³ If $\mathbf{u} = (u_{i-n_\ell}, \dots, u_i, \dots, u_{i+n_r})^T \in \mathbb{R}^m$ then $\|\mathbf{u}\|_2 = \left(\sum_{j=i-n_\ell}^{i+n_r} u_j^2 \right)^{1/2}$.

$$\begin{bmatrix} \text{yn} = \\ 0.2782 & 0.9712 & 0.2841 & 0.2118 & 0.0561 \end{bmatrix}$$

©Comment: The values of yn will normally be a little bit different since we have added random values. ☺

7. Write a function `vdm.m` (for Vandermonde) that to a given vector $\mathbf{x} = [x_1, \dots, x_m]^T \in \mathbb{R}^m$ and $n \in \mathbb{N}$ computes $\mathbf{V} \in \mathbb{R}^{m \times n}$ with elements $v_{i,j} = x_i^{j-1}$, $i = 1, \dots, m$, $j = 1, \dots, n$. You should only use one loop.
8. Write a function `gol1.m` that to given n_ℓ, n_r, n uses the `vdm` function to output the matrix \mathbf{A} of Question 2. Test with $n_\ell = 4$, $n_r = 5$ and $n = 5$.

```
>> A=gol1(4,5,5)
A =
    1     -4      16     -64     256
    1     -3       9     -27      81
    1     -2       4      -8      16
    1     -1       1      -1       1
    1      0       0       0       0
    1      1       1       1       1
    1      2       4       8      16
    1      3       9      27      81
    1      4      16      64     256
    1      5      25     125     625
```

9. Complete `gol1.m` into a function `gol2.m` that to given n_ℓ, n_r, n output $\mathbf{c} = [c_1, \dots, c_n]^T$. Test with $n_\ell = 4$, $n_r = 5$ and $n = 5$.

```
>> d=gol2(4,5,5)
d =
    -0.0000
    -0.0583
     0.0874
     0.2622
     0.3543
     0.3147
     0.1573
    -0.0408
    -0.1399
     0.0629
```

10. Write a program `gsfilter.m` with given n_ℓ, n_r, n and p . It calls `gol2.m` to compute the Savitsky-Golay coefficients \mathbf{c} . Secondly, it computes \mathbf{x} , the initial signal \mathbf{y} , and the noisy signal \mathbf{yn} using `signal2.m`. Thirdly, it computes the smooth signal \mathbf{ys} . Finally, it plots four figures: the initial signal, the noisy signal, the smooth signal and superposition of the initial and smooth signals. Test using $n_\ell = n_r = 20$, $n = 5$, and $p = 1000$ (Fig. 10.8). Try also some other values of n_ℓ, n_r, n to find the best ones.

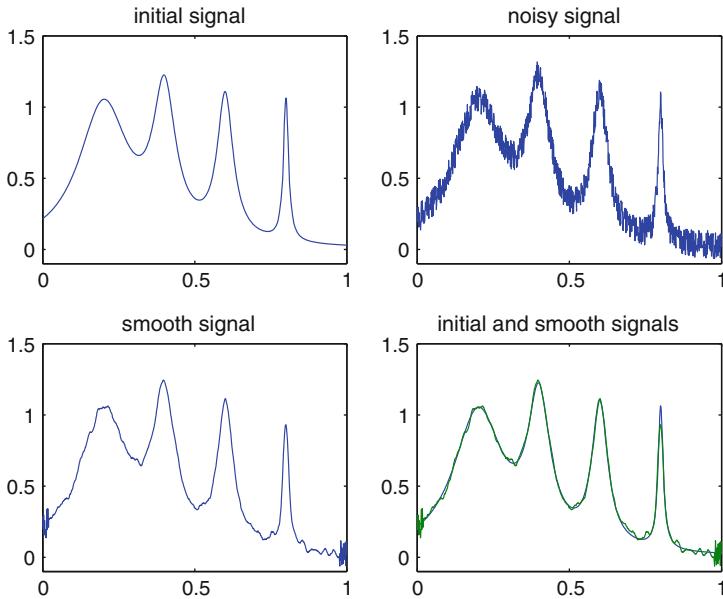


Fig. 10.8 Example of Golay-Savitsy filter

10.5 Ordinary Differential Equation and Least Squares

Exercise 10.14. Let $f \in C([0, 1])$. The problem is given by:

$$\text{Find } u(x) \text{ such that } \begin{cases} -u''(x) + u(x) = f(x) \text{ for } x \in [0, 1], \\ u(0) = u(1) = 0. \end{cases}$$

We assume that the problem has a unique solution (see Chap. 13) and we want a polynomial approximation $p(x)$ of the solution $u(x)$. We start with a uniform partition of $(0, 1)$: $x_i = ih$ with $h = \frac{1}{k+1}$ and $i = 1, \dots, k$, then we minimize

$$J(p) = p(0)^2 + p(1)^2 + \sum_{i=1}^k (-p''(x_i) + p(x_i) - f(x_i))^2 \quad (10.5)$$

If $p(x) = \sum_{j=1}^n \alpha_j \varphi_j(x)$ where $\varphi_j(x) = x^{j-1}$, then

$$\begin{aligned} p(0) &= \sum_{j=1}^n \alpha_j \varphi_j(0) = \alpha_1, & p(1) &= \sum_{j=1}^n \alpha_j, \\ p''(x_i) &= \sum_{j=3}^n \alpha_j (j-1)(j-2)x_i^{j-3}, & p(x_i) &= \sum_{j=1}^n \alpha_j x_i^{j-1}. \end{aligned}$$

1. Show that the minimization problem can be written $\min_{\alpha} \|\mathbf{A}\alpha - \mathbf{b}\|^2$, where $\mathbf{A} \in \mathbb{R}^{m \times n}$ with $m = k + 2$ and

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \dots & & 0 & & \dots & 0 \\ \vdots & & & & & & & \\ 1 & & -(j-1)(j-2)x_i^{j-3} + x_i^{j-1} & \dots & & & & \\ \vdots & & & & & & & \\ 1 & 1 & \dots & & 1 & & \dots & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 \\ f(x_1) \\ \vdots \\ f(x_k) \\ 0 \end{bmatrix}.$$

2. **Programs:** Program a function `f.m` with $f(x) = e^x$.

```
>> f([1:3]')
ans =
    2.7183
    7.3891
   20.0855
```

3. For a given k , construct the array $\mathbf{x} = [x_i = ih]_{1 \leq i \leq k}$ then construct the matrix

$$\mathbf{A}_1 = \begin{bmatrix} 1 & \dots & & \dots & \dots \\ 1 & \dots & -(j-1)(j-2)x_i^{j-3} + x_i^{j-1} & \dots \\ 1 & \dots & & \dots & \dots \end{bmatrix} \in \mathbb{R}^{k,n} \text{ with one loop}$$

and finally the matrix \mathbf{A} when adding the first and last rows. Save as `ODE1.m`. Test with $n = 4, k = 6$.

```
>> ODE1
A =
    1.0000      0      0      0
    1.0000    0.1429   -1.9796   -0.8542
    1.0000    0.2857   -1.9184   -1.6910
    1.0000    0.4286   -1.8163   -2.4927
    1.0000    0.5714   -1.6735   -3.2420
    1.0000    0.7143   -1.4898   -3.9213
    1.0000    0.8571   -1.2653   -4.5131
    1.0000    1.0000    1.0000    1.0000
```

4. Add the construction of \mathbf{b} and the computation of α while solving the system $\mathbf{A}^T \mathbf{A}\alpha = \mathbf{A}^T \mathbf{b}$. Save as `ODE2.m`. Test with $n = 4, k = 6$. Save as `ODE2.m`.

```
>> ODE2
alpha =
    0.0056
    0.6355
   -0.3626
   -0.2728
```

5. Compute and plot the polynomial p on the interval $[0, 1]$. Compare $p(x)$ to the exact solution which is $u(x) = \frac{e}{2(e-1/e)}(e^x - e^{-x}) - \frac{x}{2}e^x$. Save as `ODE3.m`. Test with $n = 4, k = 6$. See Fig. 10.9

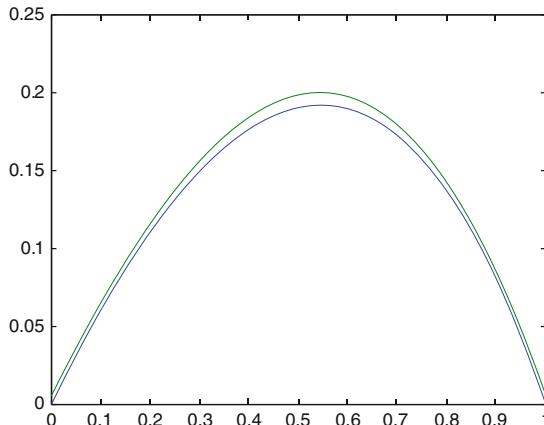


Fig. 10.9 Exact Solution and Approximation

6. Compute $\text{error} = \|p - \text{solexact}\|_\infty$. In fact, you compute $\max_{k=0,\dots,1000} |p(t_k) - u(t_k)|$ where $t_k = k/1000$. Save as ODE4.m. Test with $n = 4, m = 6$.

```
>> ODE4
error =
0.0082
```

7. What happens when n grows? Test with $n = k + 1$ and an array $\text{tabk} = [10, 20, 35, 50, 75]$ of values of k . Save as ODE5.m. Remarks?

```
>> ODE5
tabk =
    10      20      35      50      75

taberror =
1.0e-008 *
0.0019      0.6601      0.2302      0.0557      0.0804
```

8. Find a function $f1$ such that the exact solution of the equation is $u(x) = \sin(\pi x)$. Save as ODE6.m with the construction of f1.m. Test with different values of n .

10.6 Solutions

10.6.1 Orthogonal Projections and Orthogonal Sums

Exercise 10.1

1. Define b_{\max} by (10.1). By orthogonality for $j = 1, \dots, k$

$$\langle \mathbf{b}_{\max}, \mathbf{v}_j \rangle = \sum_{i=1}^k \frac{\langle \mathbf{b}, \mathbf{v}_i \rangle}{\langle \mathbf{v}_i, \mathbf{v}_i \rangle} \langle \mathbf{v}_i, \mathbf{v}_j \rangle = \frac{\langle \mathbf{b}, \mathbf{v}_j \rangle}{\langle \mathbf{v}_j, \mathbf{v}_j \rangle} \langle \mathbf{v}_j, \mathbf{v}_j \rangle = \langle \mathbf{b}, \mathbf{v}_j \rangle$$

so that by linearity $\langle \mathbf{b} - \mathbf{b}_{\max}, \mathbf{v}_j \rangle = 0$. But then $\langle \mathbf{b} - \mathbf{b}_{\max}, \mathbf{f} \rangle = 0$ for all $\mathbf{f} \in \text{max}$ since $\mathbf{f} = \sum_{i=1}^k f_i \mathbf{v}_i$. This shows existence of a \mathbf{b}_{\max} satisfying (10.2).

2. For uniqueness suppose $\mathbf{f}_1, \mathbf{f}_2 \in \text{max}$ and $\langle \mathbf{b} - \mathbf{f}_1, \mathbf{f} \rangle = \langle \mathbf{b} - \mathbf{f}_2, \mathbf{f} \rangle = 0$ for all $\mathbf{f} \in \text{max}$. Then $\langle \mathbf{b} - \mathbf{f}_1, \mathbf{f} \rangle - \langle \mathbf{b} - \mathbf{f}_2, \mathbf{f} \rangle = \langle \mathbf{f}_2 - \mathbf{f}_1, \mathbf{f} \rangle = 0$ for all $\mathbf{f} \in \text{max}$ and in particular $\langle \mathbf{f}_2 - \mathbf{f}_1, \mathbf{f}_2 - \mathbf{f}_1 \rangle = 0$ which implies that $\mathbf{f}_2 - \mathbf{f}_1 = \mathbf{0}$ or $\mathbf{f}_1 = \mathbf{f}_2$.
3. The proof that (10.2) implies (10.1) follows by uniqueness since (10.2) still holds if we replace \mathbf{b}_{\max} in (10.2) by the right hand side of (10.1).

Exercise 10.2

For the column space we use (10.1) with the columns $\mathbf{v}_1 := \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and $\mathbf{v}_2 := \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$ of \mathbf{A} . Since these vectors are orthogonal they constitute an orthogonal basis for the column space of \mathbf{A} . We find

$$\mathbf{b}_{\max} := \frac{\langle \mathbf{b}, \mathbf{v}_1 \rangle}{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle} \mathbf{v}_1 + \frac{\langle \mathbf{b}, \mathbf{v}_2 \rangle}{\langle \mathbf{v}_2, \mathbf{v}_2 \rangle} \mathbf{v}_2 = \frac{6}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + \frac{-3}{6} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 3 \\ 6 \\ 3 \end{bmatrix}.$$

A basis for the null space of \mathbf{A}^T is $\mathbf{v}_3 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$. By (10.1)

$$\mathbf{b}_{\mathcal{G}} := \frac{\langle \mathbf{b}, \mathbf{v}_3 \rangle}{\langle \mathbf{v}_3, \mathbf{v}_3 \rangle} \mathbf{v}_3 = \frac{1}{2} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}. \text{ Observe that } \mathbf{b} = \mathbf{b}_{\max} + \mathbf{b}_{\mathcal{G}}.$$

Exercise 10.3

1. The trigonometric sum formulas $\cos(A + B) = \cos A \cos B - \sin A \sin B$ and $\sin(A + B) = \sin A \cos B + \cos A \sin B$ give

$$2 \cos A \cos B = \cos(A + B) + \cos(A - B),$$

$$2 \sin A \sin B = \cos(A - B) - \cos(A + B),$$

$$2 \sin A \cos B = \sin(A + B) + \sin(A - B).$$

It follows that

$$\int_{-\pi}^{\pi} 1^2 dx = 2\pi, \int_{-\pi}^{\pi} (\cos kx)^2 dx = \int_{-\pi}^{\pi} (\sin kx)^2 dx = \pi,$$

$$\int_{-\pi}^{\pi} (\cos jx)(\sin kx) dx = 0, j, k = 1, \dots, n$$

$$\int_{-\pi}^{\pi} (\cos jx)(\cos kx) dx = \int_{-\pi}^{\pi} (\sin jx)(\sin kx) dx = 0, j, k = 1, \dots, n, j \neq k.$$

Thus $\{1, \cos x, \sin x, \dots, \cos nx, \sin nx\}$ are orthogonal functions and therefore linearly independent. Since it is a spanning set it constitutes a basis.

2. If $f \in C[-\pi, \pi]$ then the orthogonal projection is

$$f_{\max} = \frac{a_0}{2} + \sum_{k=1}^n (a_k \cos kx + b_k \sin kx), \quad (10.6)$$

where

$$a_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx \, dx, \quad b_k = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx \, dx, \quad k = 0, 1, \dots, n. \quad (10.7)$$

f_{\max} is the truncated Fourier series of f .

Exercise 10.4

1. If $\mathbf{h} \in \max \cap \mathcal{G}$ then \mathbf{h} is orthogonal to itself, $\langle \mathbf{h}, \mathbf{h} \rangle = 0$, which implies that $\mathbf{h} = 0$.
2. Suppose that we have two sums for the same \mathbf{h} , say $\mathbf{h} = \mathbf{f}_1 + \mathbf{g}_1 = \mathbf{f}_2 + \mathbf{g}_2$, then $\mathbf{f}_1 - \mathbf{f}_2 = \mathbf{g}_2 - \mathbf{g}_1$. But then $\mathbf{f}_1 - \mathbf{f}_2 \in \max \cap \mathcal{G}$ so that $\mathbf{f}_1 - \mathbf{f}_2 = 0$ and $\mathbf{g}_2 - \mathbf{g}_1 = \mathbf{f}_1 - \mathbf{f}_2 = 0$.
3. Expanding the inner product and observing that $\langle \mathbf{f}, \mathbf{g} \rangle = \langle \mathbf{g}, \mathbf{f} \rangle = 0$

$$\begin{aligned} \|\mathbf{f} + \mathbf{g}\|_2^2 &= \langle \mathbf{f} + \mathbf{g}, \mathbf{f} + \mathbf{g} \rangle = \langle \mathbf{f}, \mathbf{f} \rangle + \langle \mathbf{f}, \mathbf{g} \rangle + \langle \mathbf{g}, \mathbf{f} \rangle + \langle \mathbf{g}, \mathbf{g} \rangle \\ &= \|\mathbf{f}\|_2^2 + \|\mathbf{g}\|_2^2. \end{aligned}$$

Exercise 10.5

1. If $\mathbf{f} \in \text{span}(\mathbf{A})$ and $\mathbf{g} \in \ker(\mathbf{A}^T)$ then $\mathbf{f} = \mathbf{Ax}$ for some $\mathbf{x} \in \mathbb{C}^n$ and $\mathbf{A}^T \mathbf{g} = \mathbf{0}$. But then $\mathbf{f}^T \mathbf{g} = \mathbf{x}^T \mathbf{A}^T \mathbf{g} = \mathbf{0}$ and $\text{span}(\mathbf{A}) + \ker(\mathbf{A}^T)$ is an orthogonal sum.
2. Suppose $\mathbf{b}_1 = \mathbf{b}_{\max}$ is the orthogonal projection onto $\text{span}(\mathbf{A})$ of some $\mathbf{b} \in \mathbb{R}^m$. By (10.2) $\mathbf{f}^T (\mathbf{b} - \mathbf{b}_1) = 0$ for all $\mathbf{f} \in \text{span}(\mathbf{A})$ and since $\mathbf{f} = \mathbf{Ax}$ for some \mathbf{x} it follows that $\mathbf{x}^T \mathbf{A}^T (\mathbf{b} - \mathbf{b}_1) = 0$ for all $\mathbf{x} \in \mathbb{R}^n$. Then $\mathbf{A}^T \mathbf{b}_2 = \mathbf{0}$, where $\mathbf{b}_2 := \mathbf{b} - \mathbf{b}_1$. Thus $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where $\mathbf{b}_1 \in \text{span}(\mathbf{A})$ and $\mathbf{b}_2 \in \ker(\mathbf{A}^T)$.

Exercise 10.6

1. Clearly $\mathbf{A}^T = [3, -1]$ so $\mathbf{A} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$.
2. $\mathbf{b}_1 = \frac{\mathbf{b}^T \mathbf{v}_1}{\mathbf{v}_1^T \mathbf{v}_1} \mathbf{v}_1 = \frac{[5 \ 5] \begin{bmatrix} 3 \\ -1 \end{bmatrix}}{3^2 + 1^2} \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$. $\mathbf{b}_2 = \mathbf{b} - \mathbf{b}_1 = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$.

Exercise 10.7

1. This follows since $\max \oplus \mathcal{G}$ is an orthogonal sum and $\mathbf{Ax} - \mathbf{b}_1 \in \max$ and $\mathbf{b}_2 \in \mathcal{G}$.

2. By Pythagoras $\|(\mathbf{Ax} - \mathbf{b}_1) + \mathbf{b}_2\|_2^2 = \|\mathbf{Ax} - \mathbf{b}_1\|_2^2 + \|\mathbf{b}_2\|_2^2 \geq \|\mathbf{b}_2\|_2^2$ for any $\mathbf{x} \in \mathbb{R}^n$. Any \mathbf{x} such that $\mathbf{Ax} = \mathbf{b}_1 \in \text{span}(\mathbf{A})$ minimizes $\|\mathbf{Ax} - \mathbf{b}\|_2^2$ and is therefore a least squares solution.
3. If \mathbf{x} is a least squares solution then $\mathbf{b}_1 = \mathbf{Ax}$. This means that \mathbf{b}_1 is a linear combination of the columns of \mathbf{A} . This representation is unique if and only if \mathbf{A} has linearly independent columns and this happens if and only if $\text{rank}(\mathbf{A}) = n$.
4. Suppose $\mathbf{b}_1 - \mathbf{Ax} = \mathbf{0}$. Since $\mathbf{A}^T \mathbf{b}_2 = \mathbf{0}$ then $\mathbf{0} = \mathbf{A}^T(\mathbf{b}_1 - \mathbf{Ax} + \mathbf{b}_2) = \mathbf{A}^T(\mathbf{b} - \mathbf{Ax})$. Thus \mathbf{x} satisfies the normal equations.
5. Suppose $\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$. Then $\mathbf{A}^T(\mathbf{b} - \mathbf{Ax}) = \mathbf{0}$ or $\mathbf{A}^T(\mathbf{b}_1 + \mathbf{b}_2 - \mathbf{Ax}) = \mathbf{0}$. Since $\mathbf{A}^T \mathbf{b}_2 = \mathbf{0}$ it follows that $\mathbf{A}^T(\mathbf{b}_1 - \mathbf{Ax}) = \mathbf{0}$. But then $\mathbf{b}_1 - \mathbf{Ax} \in \text{max} \cap \mathcal{G}$ which implies that $\mathbf{b}_1 - \mathbf{Ax} = \mathbf{0}$.
6. Recall that the matrix \mathbf{B} is symmetric and positive definite if $\mathbf{B} = \mathbf{B}^T$ and $\boldsymbol{\alpha}^T \mathbf{B} \boldsymbol{\alpha} \geq 0$ for any $\boldsymbol{\alpha} \in \mathbb{R}^n$, and $\boldsymbol{\alpha}^T \mathbf{B} \boldsymbol{\alpha} = 0$ implies $\boldsymbol{\alpha} = \mathbf{0}$. Moreover, a symmetric and positive definite matrix is nonsingular. The matrix $\mathbf{B} = \mathbf{A}^T \mathbf{A}$ is symmetric since $\mathbf{B}^T = (\mathbf{A}^T \mathbf{A})^T = \mathbf{B}$. For any $\boldsymbol{\alpha} \in \mathbb{R}^n$, we find $\boldsymbol{\alpha}^T \mathbf{B} \boldsymbol{\alpha} = \boldsymbol{\alpha}^T \mathbf{A}^T \mathbf{A} \boldsymbol{\alpha} = \mathbf{z}^T \mathbf{z}$, where $\mathbf{z} := \mathbf{A} \boldsymbol{\alpha}$. Thus $\boldsymbol{\alpha}^T \mathbf{B} \boldsymbol{\alpha} \geq 0$, and if $\boldsymbol{\alpha}^T \mathbf{B} \boldsymbol{\alpha} = 0$ then $\mathbf{z} = \mathbf{A} \boldsymbol{\alpha} = \mathbf{0}$ and then $\boldsymbol{\alpha} = \mathbf{0}$ if and only if $\text{rank}(\mathbf{A}) = n$.
7. $\mathbf{b}_1 = \mathbf{Ax} = \mathbf{A} \mathbf{A}^\dagger \mathbf{b}$.
8. $\mathbf{A}^\dagger \mathbf{A} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A} = \mathbf{I}$. Therefore, $\mathbf{A}^{-1} = \mathbf{A}^\dagger$ since the inverse is unique.

Exercise 10.8

1. $\mathcal{G} \neq \emptyset$ since $\mathbf{0} \in \mathcal{G}$. Then if $\mathbf{g}_1, \mathbf{g}_2 \in \mathcal{G}$, and $\lambda_1, \lambda_2 \in \mathbb{R}$, for all $\mathbf{f} \in \text{max}$, we have

$$\langle \lambda_1 \mathbf{g}_1 + \lambda_2 \mathbf{g}_2, \mathbf{f} \rangle = \lambda_1 \langle \mathbf{g}_1, \mathbf{f} \rangle + \lambda_2 \langle \mathbf{g}_2, \mathbf{f} \rangle = 0$$

thus $\lambda_1 \mathbf{g}_1 + \lambda_2 \mathbf{g}_2 \in \mathcal{G}$. We deduce that \mathcal{G} is a subspace of \mathcal{E} .

2. If max is a finite dimension subspace, for $\mathbf{b} \in \text{max}$, let $\mathbf{f} = \mathbf{b}_{\text{max}}$ be the orthogonal projection of \mathbf{b} on max and define $\mathbf{g} = \mathbf{e} - \mathbf{f}$. It is easy to check that $\mathbf{g} \in \mathcal{G}$.

10.6.2 Curve Fitting by Least Squares

Exercise 10.9

1. It follows immediately from the condition given that $\mathbf{A} \boldsymbol{\alpha} = \mathbf{0} \implies \boldsymbol{\alpha} = \mathbf{0}$ and \mathbf{A} has rank n .
2. Suppose $\varphi_j(x) = x^{j-1}$ for $j = 1, \dots, n$. If $\boldsymbol{\alpha} \in \mathbb{R}^n$ is nonzero then the function $\phi_{\boldsymbol{\alpha}}$ is a nonzero polynomial of degree at most $n-1$ and as such has at most $n-1$ distinct zeros. Thus it cannot vanish at n distinct points and we conclude that $\boldsymbol{\alpha} = \mathbf{0}$.

Exercise 10.10

1. The vector $\boldsymbol{\alpha} = (\alpha_i)$ is the solution of $\mathbf{A}^T \mathbf{A} \boldsymbol{\alpha} = \mathbf{A}^T \mathbf{y}$, where

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \end{bmatrix} = \begin{bmatrix} 1 & -2 \\ 1 & -1 \\ 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \end{bmatrix}$$

for the line and the parabola, respectively.

For the line, this gives $\begin{bmatrix} 5 & 0 \\ 0 & 10 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} = \begin{bmatrix} 15 \\ 20 \end{bmatrix}$ hence $\alpha_1 = 3$ et $\alpha_2 = 2$.

For the parabola, the system is $\begin{bmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{bmatrix} = \begin{bmatrix} 15 \\ 20 \\ 44 \end{bmatrix}$; the solution is

$\alpha_1 = 1, \alpha_2 = 2$ et $\alpha_3 = 1$.

2. Program:

```
x=-2:2;
y=[3/2, -1, 1, 5, 17/2];
disp('components for the straight line')
a=polyfit(x,y,1)
disp('components for the parabola')
b=polyfit(x,y,2)
t=-2:0.01:2;
yd=polyval(a,t);
yp=polyval(b,t);
subplot(1,2,1)
plot(x,y,'x',t,yd);
subplot(1,2,2)
plot(x,y,'x',t,yp);
```

Exercise 10.11

1. $J(\boldsymbol{\alpha}) = \|\mathbf{A}\boldsymbol{\alpha} - \mathbf{b}\|_2^2$ where $a_{ij} = \varphi_j(x_i, y_i)$, $i = 1, \dots, m$ and $j = 1, \dots, 5$ and $b_i = \varphi_6(x_i, y_i) = 1$, $i = 1, \dots, m$.
2. For the given example, $\mathbf{A} \in \mathbb{R}^{8 \times 5}$ is defined by:

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 7 - 4\sqrt{3} & \frac{1}{4} & 1 - \frac{\sqrt{3}}{2} & 2 - \sqrt{3} & \frac{1}{2} \\ 4 & 0 & 1 + \frac{\sqrt{3}}{2} & 2 & 0 \\ 7 + 4\sqrt{3} & \frac{1}{4} & 3 + \frac{3\sqrt{3}}{2} & 2 + \sqrt{3} & \frac{1}{2} \\ 16 & 1 & 4 & 4 & 1 \\ 7 + 4\sqrt{3} & \frac{9}{4} & \frac{3}{2}(2 + \sqrt{3}) & 2 + \sqrt{3} & \frac{3}{2} \\ 4 & 4 & 4 & 2 & \frac{3}{2} \\ 7 - 4\sqrt{3} & \frac{9}{4} & 3 - \frac{3\sqrt{3}}{2} & 2 - \sqrt{3} & \frac{3}{2} \end{bmatrix}.$$

3. mat.m

```
function M=mat(X,Y);
M(:,1)=X.*X;
M(:,2)=Y.*Y;
M(:,3)=X.*Y;
M(:,4)=X;
M(:,5)=Y;
M(:,6)=ones(1,length(X));
```

4. conic1.m

```
X=[1,-1,2]';
Y=[0,3,4]';

M=mat(X,Y)

A=M(:,1:5)
b=M(:,6)
```

5. phi.m

```
function Z = phi(X,Y,a)
Z=a(1)*X.*X+a(2)*Y.*Y+a(3)*X.*Y+a(4)*X...
+a(5)*Y+a(6);
```

6. (and 7,8). conic2.m

```
p=[0,1;2-sqrt(3),1/2;2,0;2+sqrt(3),1/2;...
4,1;2+sqrt(3),3/2;2,2;2-sqrt(3),3/2];
x=p(:,1); y=p(:,2);

%{
n=10;
theta=2*pi*rand(n,1);
x=1.8+2*cos(theta)+0.6*rand(n,1);
y=0.8+sin(theta)+0.6*rand(n,1);
%}

A=mat(x,y);
b=-ones(length(x),1);
beta=(A'*A)\(A'*b)
alpha=[beta;1]

[Xg,Yg]=meshgrid([-1:0.1:5],[-0.5:0.1:2.5]);
Zg=phi(Xg,Yg,alpha);
contour(Xg,Yg,Zg,[0 0], 'b')
hold on
plot(x,y,'*r')
hold off
```

10.6.3 Periodic Signal

Exercise 10.12

1.2.3. The functions `SP1.m` and `SP2.m` are contained in the first few lines of `SP3.m`

```
function SP3(m,n);
[x,yi,yb]=signal1(m);
A=sin(x*(1:n)*pi);
alpha=(A'*A)\(A'*yb)
y1=A*alpha;
subplot(2,2,1)
plot(x,yi)
axis([-1 1 -1.6 1.6])
title('initial signal')
subplot(2,2,2)
plot(x,yb)
axis([-1 1 -1.6 1.6])
title('noisy signal')
subplot(2,2,3)
plot(x,y1)
axis([-1 1 -1.6 1.6])
title('smooth signal')
subplot(2,2,4)
plot(x,yi,x,y1)
axis([-1 1 -1.6 1.6])
title('initial and smooth signals')
```

②Comment: The matrix (jx_i) is constructed without loops using the matrix product with command `x*(1:n)`. ④

The noisy signal is $\mathbf{yn} = \mathbf{y} + noise = \mathbf{ys} + \mathbf{g}$ where $\mathbf{ys} \in \text{span}(\mathbf{A})$ and $\mathbf{g} \in \ker(\mathbf{A}^T)$. We look for $\mathbf{y} \simeq \mathbf{ys}$ and $noise \simeq \mathbf{g}$. Most of the noise is in the orthogonal complement $\ker(\mathbf{A}^T)$ if there is enough elements in the basis. If n is too large, a significant part of the noise is still in the row space of \mathbf{A} and if n is too small ($n < 6$ see the construction of \mathbf{y} in `signal1`) a significant part of the signal \mathbf{y} is in $\ker(\mathbf{A}^T)$.

4. `SP4.m`

```
function SP4(m,N);
[x,yi,yb]=signal2(m);
A1=cos(x*(0:N)*pi);
A2=sin(x*(1:N)*pi);
A(:,1:2:2*N+1)=A1;
A(:,2:2:2*N)=A2;
alpha=(A'*A)\(A'*yb)
y1=A*alpha;
subplot(2,2,1)
plot(x,yi)
axis([-1 1 -2 2.7])
title('initial signal')
subplot(2,2,2)
plot(x,yb)
```

```

axis([-1 1 -2 2.7])
title('noisy signal')
subplot(2,2,3)
plot(x,y1)
axis([-1 1 -2 2.7])
title('smooth signal')
subplot(2,2,4)
plot(x,yi,x,y1)
axis([-1 1 -2 2.7])
title('initial and smooth signals')

```

Comment: Again the construction of the matrix A is done without using loops. The command $A(:, 1:2:2*N+1)$ (respectively $A(:, 2:2:2*N)$) is used to obtain odd columns (even columns). ☺

10.6.4 Savitsky-Golay Filter

Exercise 10.13

- The number of functions ϕ_j is n which is the dimension of \mathbb{P}_{n-1} so that, it is sufficient to prove that the functions $\varphi_j(x) := \left(\frac{x - x_k}{\Delta x}\right)^{j-1}$ are linearly independent to conclude that we have a basis.

For any ℓ from 0 to $n-1$, the ℓ -th derivative of ϕ_j at x_k is 0 if $j \neq \ell + 1$. So that if $\sum_{j=1}^n \lambda_j \varphi_j(x) = 0$, when computing the ℓ -th derivative at x_k , we obtain $\lambda_{j_\ell} \frac{j_\ell!}{(\Delta x)^{j_\ell}} = 0$ with $j_\ell = \ell + 1$ and we deduce that $\lambda_{j_\ell} = 0$. Thus we obtain the basis.

- Since

$$a_{i,j} = \varphi_j(x_{k+i-n_\ell-1}) = \left(\frac{x_{k+i-n_\ell-1} - x_k}{\Delta x}\right)^{j-1} = (i - n_\ell - 1)^{j-1},$$

we obtain that

$$A = \begin{bmatrix} 1 & -n_\ell & \dots & (-n_\ell)^{n-2} & (-n_\ell)^{n-1} \\ 1 & -n_\ell + 1 & \dots & (-n_\ell + 1)^{n-2} & (-n_\ell + 1)^{n-1} \\ \vdots & & & \vdots & \\ 1 & -1 & \dots & (-1)^{n-2} & (-1)^{n-1} \\ 1 & 0 & \dots & \dots & 0 \\ 1 & 1 & \dots & 1 & 1 \\ \vdots & & & \vdots & \\ 1 & n_r - 1 & \dots & (n_r - 1)^{n-2} & (n_r - 1)^{n-1} \\ 1 & n_r & \dots & n_r^{n-2} & n_r^{n-1} \end{bmatrix}.$$

Due to the choice of the polynomial basis, you can notice that A is independent of k and the point x_k .

3. A is a Vandermonde matrix so that it is of rank n .
4. The minimizing polynomial has components α such that $A^T A \alpha = A^T b$ with $b_i = f_{k+i-n_\ell-1}$. Thus, $\alpha = (A^T A)^{-1} A^T b$. But $g_k = p_k(x_k) = \sum_{j=1}^n \alpha_j \varphi_j(x_k) = \alpha_1$. Then the $g_k = e_1^T \alpha$ where $e_1 = [1, 0, \dots, 0]^T$ is the first vector of the canonic basis of \mathbb{R}^n . Finally, $g_k = c^T b = \sum_{i=-n_\ell}^{n_r} c_i f_{k+i}$ where $c = (e_1^T (A^T A)^{-1} A^T)^T = A (A^T A)^{-1} e_1$.

5,6. signal2.m

```
function [x, yi, yb]=signal2(n);
h=1/(n-1);
x=0:h:1;
yi=1./(1+100*(x-0.2).^2)+1./(1+500*(x-0.4).^2)...
+1./(1+1000*(x-0.6).^2)+1./(1+10000*(x-0.8).^2);
V=(-1+2*rand(1,n));
yb=V*0.1+yi;
```

7. vdm.m

```
function A=vdm(x,n)
m=length(x);
A=ones(m,n);
for j=1:n-1
    A(:, j+1)=A(:, j).*x;
end;
```

Comment: We have used a loop to build the matrix A but you can construct two matrices and use the powers instead. ☺

8,9. gol2.m

```
function c= gol2(nl ,nr ,n)
x=(-nl : nr )';
A=vdm(x,n);
e=zeros(n,1);
e(1)=1;
c=A*inv(A'*A)*e;
```

10.gsfiltter.m

```
NL=20;NR=25;
m=5;
% computation of c
c=gol2(NL,NR,m);
p=1000;
[x, yi, yn]=signal2(p);
% Filtering
ys=yn;
for i=NL+1:p-NR
    ys(i)=c'*yn(i-NL: i+NR)';
end;
% Graphs
subplot(2 ,2 ,1)
```

```

plot(x,yi)
axis([0 1 -0.1 1.5])
title('initial signal')
subplot(2,2,2)
plot(x,yn)
axis([0 1 -0.1 1.5])
title('noisy signal')
subplot(2,2,3)
plot(x,ys)
axis([0 1 -0.1 1.5])
title('smooth signal')
subplot(2,2,4)
plot(x,yi,x,ys)
axis([0 1 -0.1 1.5])
title('initial and smooth signals')

```

10.6.5 Ordinary Differential Equation and Least Square

Exercise 10.14

ODE4.m

```

n=4;
k=6;
h=1/(k+1);
x=(h:h:1-h)';
for j=1:n
    A2(:,j)=-(j-1)*(j-2)*x.^^(j-3)+x.^(j-1);
end
A=[zeros(1,n);A2;ones(1,n)];
A(1,1)=1
b=[0;f(x);0];
alpha=(A'*A)\(A'*b)
t=0:1/1000:1;
e=exp(1);
uex=e/(2*(e-1/e))*(exp(t)-exp(-t))-t/2.*exp(t);
uapp=polyval(alpha(n:-1:1),t);
plot(t,uex,t,uapp)
error=norm(uex-uapp,inf)

```

Comment: To compute $t \times e^t$, where t is an array, the MATLAB instruction is $t.*\exp(t)$. For details on the difference between $*$ and $.*$, see Chap. 1.

ODE5.m

```

tabk=[10, 15, 20, 35, 50, 75];
for i=1:length(tabk)
    k=tabk(i)
    n=k+1;
    h=1/(k+1);
    x=(h:h:1-h)';
    A1=x*ones(1,n);

```

```

for j=1:n
    A2(:,j)=-(j-1)*(j-2)*A1(:,j).^(j-3)...
        +A1(:,j).^(j-1);
end
A=[zeros(1,n);A2;ones(1,n)];
A(1,1)=1;
b=[0;f(x);0];
alpha=(A'*A)\(A'*b);
t=0:1/1000:1;
e=exp(1);
uex=e/(2*(e-1/e))*(exp(t)-exp(-t))-t/2.*exp(t);
uapp=polyval(alpha(n:-1:1),t);
taberror(i)=norm(uex-uapp,inf);
end;
tabk
taberror

```

② **Comment:** The error is almost constant or increasing. The matrix $A^T A$ is very badly conditionned (see Chap. 4) when n increases. One answer to this problem could be the choice of an optimal polynomial basis.

To conclude, another example is given by $f_1(x) = (\pi^2 + 1) \sin(\pi x)$. ③

Chapter 11

Continuous and Discrete Approximations

In Chap. 10, we studied approximations using 2-norms that are defined from a dot product, and orthogonality was heavily used. In the first section of this chapter, we study polynomial approximation of continuous functions on an interval $[a, b]$ using the **uniform norm** $\|\cdot\|_\infty$. The **Chebyshev Polynomials** are studied in Exercise 11.1 and are used in Exercise 11.2 to bound the distance from a function to \mathbb{P}_n . In Exercise 11.3 we give a proof of the **Weierstrass Theorem** by studying a sequence of polynomials that converges to a given continuous function. Moreover, in Exercise 11.4, we evaluate the rate of convergence. Then for a given degree, we consider best polynomial approximation in Exercises 11.5 and 11.6.

In the next section, we use **Fourier series** to solve the simplified **Heat equation** in Exercise 11.7 and an approximation of the exact solution is plotted.

Then we switch to discrete approximation with the theory of **Haar wavelets** in Exercise 11.8 and the corresponding programs in Exercise 11.9. This modern tool can also be used as a filter for noisy signals.

11.1 Polynomial Approximation with $\|\cdot\|_\infty$

Review:

Theorem 11.1. Weirstrass Theorem: Suppose f is a continuous real or complex-valued function defined on the real interval $[a, b]$. For every $\varepsilon > 0$, there exists a polynomial p such that for all $x \in [a, b]$, we have $|f(x) - p(x)| < \varepsilon$, or equivalently, in the uniform norm $\|f - p\|_\infty < \varepsilon$.

A constructive proof of this theorem on $[0, 1]$ using Bernstein polynomials is outlined as Exercise 11.3.

The **equioscillation theorem** attributed to Chebyshev concerns the best approximation of continuous functions by polynomials of given degree.

Theorem 11.2. Let $f \in C([a, b])$ be real valued. Among all the polynomials of degree at most n , the polynomial p_n minimizes the uniform norm of the difference $\|f - p\|_\infty$ if and only if there are $n + 2$ points $a \leq x_0 < x_1 < \dots < x_{n+1} \leq b$ such that $f(x_i) - p_n(x_i) = \sigma(-1)^i \|f - p_n\|_\infty$ where $\sigma = 1$ or $\sigma = -1$.

♦

Exercise 11.1. Chebyshev Polynomials

Let $T_0(x) := 1$, $T_1(x) := x$ and for $n \in \mathbb{N}$, $n \geq 1$, we define the polynomials recursively by

$$T_{n+1}(x) := 2xT_n(x) - T_{n-1}(x). \quad (11.1)$$

1. Show that $T_n \in \mathbb{P}_n \setminus \mathbb{P}_{n-1}$ and compute the leading coefficient of T_n . ▷ *Math Hint*¹ ◁
2. Prove that for $x \in [-1, 1]$, $T_n(x) = \cos(n \arccos x)$. ▷ *Math Hint*² ◁
3. Find the roots x_1, \dots, x_n of T_n and compute $\|\prod_{i=1}^n (\cdot - x_i)\|_{\infty, [-1, 1]}$. ▷ *Math Hint*³ ◁
4. Find the points x'_k giving the extrema of T_n on $[-1, 1]$.
5. **Programs:** Write a MATLAB function computing $T_n(\mathbf{x})$ by the recurrence relation where n is an integer and \mathbf{x} an array. Save as `chebyshevpol.m`. ▷ *MATLAB Hint*⁴ ◁ *Test*

```
>> y=chebyshevpol(5, -2:0.5:0.5)
y =
-362.0000   -61.5000    -1.0000    -0.5000         0      0.5000
```

6. Write a program to plot T_n for $n = 0, \dots, 8$. Save as `plotchebypol.m`. See Fig. 11.1.

Exercise 11.2. Distance from a function to spaces of polynomials

In this exercise, we will prove that if $f \in C^n([a, b])$ then

$$\min_{p \in \mathbb{P}_{n-1}} \|f - p\|_{\infty, [a, b]} \leq \frac{2}{n!} \left(\frac{b-a}{4} \right)^n \|f^{(n)}\|_{\infty, [a, b]} \quad (11.2)$$

For the sites $\mathbf{t} = [t_1, \dots, t_n]^T$ given by $a \leq t_1 < t_2 < \dots < t_n \leq b$, let $p_{n-1} \in \mathbb{P}_{n-1}$ be the Lagrange interpolating polynomial, i.e. $p_{n-1}(t_i) = f(t_i)$ for all i . We recall from Chap. 6 that for $t \in [a, b]$, there exists $\xi \in [a, b]$ such that the error satisfies

$$f(t) - p_{n-1}(t) = \frac{f^{(n)}(\xi)}{n!} \prod_{j=1}^n (t - t_j). \quad (11.3)$$

¹ Use induction.

² Factorize $\cos(n+1)\theta + \cos(n-1)\theta$.

³ Use the roots and the leading coefficient.

⁴ The function is calling itself.

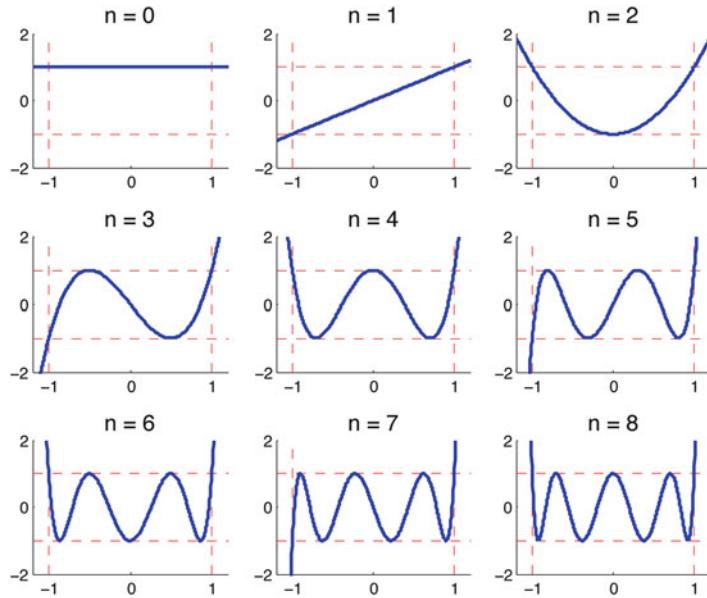


Fig. 11.1 Graphs of $T_n(x)$

see (6.1) for the proof.

1. Let x and x_i be defined by $t = \frac{a+b}{2} + \frac{b-a}{2}x$ and $t_i = \frac{a+b}{2} + \frac{b-a}{2}x_i$ for $i = 1, \dots, n$. Show that $t, t_i \in [a, b] \implies x, x_i \in [-1, 1]$.
2. Transform the right part of (11.3) using the new variables.
3. How can we choose the x_i such that $\left| \prod_{i=1}^n (x - x_i) \right| \leq 2^{-(n-1)}$ for any $x \in [-1, 1]$. ▷ Math Hint⁵ ◁
4. Deduce the result (11.2).
5. **Programs** to construct Lagrange interpolants are proposed in Exercise 6.9 where an example using the Chebyshev sites i.e. the roots of T_n is shown in Fig. 6.3.

Exercise 11.3. Weierstrass Theorem on $[0, 1]$

Karl Theodor Wilhelm Weierstrass (1815–1897) was a German mathematician, winner of the Copely Medal in 1895. He studied mathematics at the University of Münster and obtained a professorship in Berlin. He consolidated the work of Cauchy on irrational numbers and led to a new understanding. His most famous works are the elliptic functions and he was the first to give an example of a continuous function differentiable nowhere.



⁵ See Exercise 11.1.

We use the Bernstein polynomials studied in Exercise 7.2. For a non-negative integer n and the usual binomial coefficients $\binom{n}{i} := \frac{n!}{(n-i)!i!}$, we recall that the **Bernstein polynomials** are defined by

$$B_i^n(x) := \binom{n}{i} x^i (1-x)^{n-i} \text{ for } i \in \{0, \dots, n\}, \quad B_i^n(x) := 0 \text{ for } i \notin \{0, \dots, n\}.$$

See also Section 7.2.

1. Prove the following inequality and equalities:

$$B_i^n(x) \geq 0, \quad x \in [0, 1], \quad (11.4)$$

$$r_0(x) := \sum_{i=0}^n B_i^n(x) = 1, \quad (11.5)$$

$$r_1(x) := \sum_{i=0}^n i B_i^n(x) = nx, \quad (11.6)$$

$$r_2(x) := \sum_{i=0}^n i(i-1) B_i^n(x) = n(n-1)x^2, \quad (11.7)$$

$$\bar{r}_2(x) := \sum_{i=0}^n (i-nx)^2 B_i^n(x) = nx(1-x). \quad (11.8)$$

▷ *Math Hint*⁶ ◁

2. We define for $n \in \mathbb{N}$ the polynomials $p_n(x) := \sum_{i=0}^n f(i/n) B_i^n(x)$.

Let $\varepsilon > 0$. We will prove that $\|f - p_n\|_\infty < \varepsilon$ for n large enough.

The function f is continuous on the closed and bounded interval $[0, 1]$. Therefore, f is bounded, i.e., $|f| \leq M$ for some constant M . Moreover, since f is uniformly continuous there exists $\eta > 0$ such that

$$\forall x, x' \in [0, 1], |x - x'| < \eta \Rightarrow |f(x) - f(x')| < \varepsilon/2. \quad (11.9)$$

For a given $x \in [0, 1]$, we define $I_x := \{i \in \{0, \dots, n\} : |x - i/n| < \eta\}$ and $J_x := \{0, \dots, n\} \setminus I_x$. Prove that

$$\left| \sum_{i \in I_x} (f(i/n) - f(x)) B_i^n(x) \right| \leq \varepsilon/2, \quad \left| \sum_{i \in J_x} (f(i/n) - f(x)) B_i^n(x) \right| \leq \frac{M}{2\eta^2 n}$$

▷ *Math Hint*⁷ ◁

⁶ For (11.5)–(11.7), use the binomial expansion of $(u + v)^n$, and take derivatives with respect to u . Then set $u = x$ and $v = 1 - x$.

⁷ If $i \in J_x$ then $1 \leq \frac{(i-nx)^2}{n^2 \eta^2}$.

3. Deduce that $\|f - p_n\|_\infty < \varepsilon$ for n large enough.

Comment: The result on any interval $[a, b]$ is deduced by a change of variables $t = a + (b - a)x$ with $x \in [0, 1]$. \blacklozenge

Exercise 11.4. Numerical evaluation of the error and extrapolation

For $f \in C([0, 1])$ we again consider $p_n(t) = \sum_{i=0}^n f(i/n)B_i^n(t)$. For a fixed t we can compute $p_n(t)$ using the de Casteljau algorithm (see Chap. 7 and Algorithm 7.1). In our case it takes the following form

Given the $n+1$ real values $(y_i)_{i=0}^n$ with $y_i = f(i/n)$, for a given t we define the functions $y_i^r(t)$ by

1. $y_i^0(t) = y_i, i = 0, \dots, n$
2. $y_i^r(t) = (1-t)y_i^{r-1}(t) + ty_{i+1}^{r-1}(t), i = 0, \dots, n-r, r = 1, \dots, n$.

Then $p_n(t) = y_0^n(t)$.

1. Write a MATLAB function `decasteljau.m` that for given arrays $\mathbf{y} \in \mathbb{R}^{n+1}$ and $\mathbf{t} \in \mathbb{R}^N$ containing sample values $t_1, \dots, t_N \in [0, 1]$ computes $p_n(\mathbf{t})$. \triangleleft
- MATLAB Hint⁸ \triangleright

```
>> decasteljau([-1, -1/8, 1/8, 1], (0:0.2:1))
ans =
-1.0000    -0.5400    -0.1700     0.1700     0.5400     1.0000
```

2. Write a program to study the error $\|f - p_n\|_\infty$ for different values of n contained in an array `arrn`. Plot $\log(\text{error})$ as a function of $\log(n)$. Save in `errorstudy1.m`. Test with $arrn = 100 : 110$ and $f(x) = \sin(\pi x)$. The function f should be computed in a function file `f.m`. How does the rate of convergence depend on n ? See Fig. 11.2.
3. Show that if $f \in \mathbb{P}_1$ then $p_n = f$ for $n \geq 1$.
4. For $\alpha_n \in \mathbb{R}$, we define $q_n = \alpha_n p_{2n} + (1 - \alpha_n)p_n$. Show that for an appropriate choice of α_n , if $f \in \mathbb{P}_2$ then $f = q_n$ for $n \geq 2$.
5. For $\bar{\alpha}_n(t) \in \mathbb{P}_2$, we define $r_n = p_n - \bar{\alpha}_n(t)p_n''$. Show that for appropriate choices of the function $\bar{\alpha}_n$, if $f \in \mathbb{P}_2$ then $f = r_n$ for $n \geq 2$. \triangleright Math Hint⁹ \triangleleft
6. We choose α_n (respec. $\bar{\alpha}_n(x)$) giving the reproduction of \mathbb{P}_2 for q_n (respec. r_n). Write a program `errorstudy2.m` to study the errors $\|f - q_n\|$ and $\|f - r_n\|$ depending on n . Test with $arrn = 100 : 105$. How does the rate of convergence depend on n ? See Fig. 11.3.

⁸ See Exercise 7.1.

⁹ From Chap. 7, we recall that if $p_n(t) = \sum_{i=0}^n y_i B_i^n(t)$, then $p_n''(x) = \frac{n(n-1)}{2} \sum_{j=0}^{n-2} \Delta^2 y_j B_j^{n-2}(t)$ where $\Delta^2 y_j = y_{j+2} - 2y_{j+1} + y_j$.

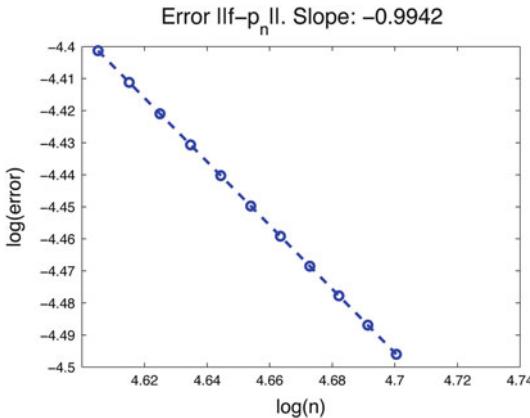


Fig. 11.2 Evaluation of the error $\|f - p_n\|$

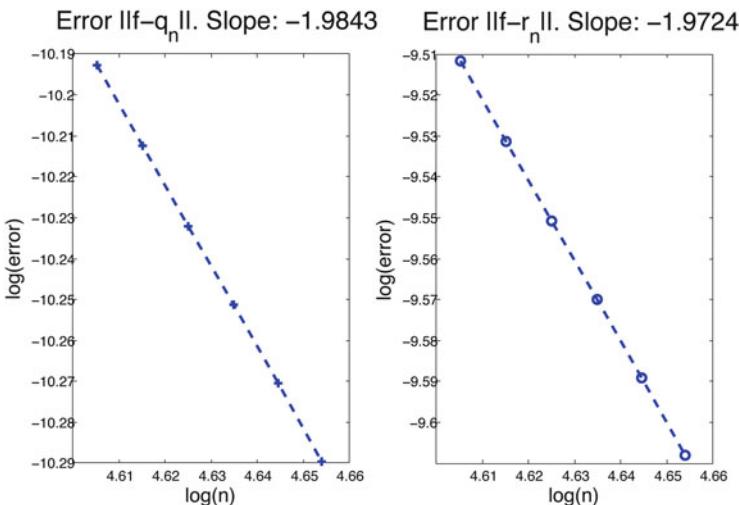


Fig. 11.3 Evaluation of the errors $\|f - q_n\|$ and $\|f - r_n\|$

Exercise 11.5. Equioscillating functions

For a continuous function f on $[0, 1]$ and sites $0 \leq x_1 < \dots < x_n < x_{n+1} \leq 1$, we wish to compute the polynomial $p(t) = \sum_{k=1}^n \alpha_k t^{k-1}$ of degree at most $n - 1$ that satisfies:

$$f(x_i) - p(x_i) = (-1)^{i+1}(f(x_1) - p(x_1)), \quad i = 2, \dots, n + 1 \quad (11.10)$$

1. Let $\alpha = [\alpha_1, \dots, \alpha_n]^T$. Deduce that $A\alpha = b$ where $A = A_1 - A_2 \in \mathbb{R}^{n \times n}$ and $b = b_1 - b_2 \in \mathbb{R}^n$ are defined by \triangleright Math Hint¹⁰ \triangleleft

$$\begin{aligned} A_1 &= \begin{bmatrix} 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \dots & x_3^{n-1} \\ 1 & x_4 & x_4^2 & \dots & x_4^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \dots & x_{n+1}^{n-1} \end{bmatrix} \\ A_2 &= \begin{bmatrix} -1 & -x_1 & -x_1^2 & \dots & -x_1^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ -1 & -x_1 & x_1^2 & \dots & -x_1^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ (-1)^n & (-1)^n x_1 & (-1)^n x_1^2 & \dots & (-1)^n x_1^{n-1} \end{bmatrix} \end{aligned} \quad (11.11)$$

and

$$b_1 = \begin{bmatrix} f(x_2) \\ f(x_3) \\ f(x_4) \\ \vdots \\ f(x_{n+1}) \end{bmatrix}, \quad b_2 = f(x_1) \begin{bmatrix} -1 \\ 1 \\ -1 \\ \vdots \\ (-1)^n \end{bmatrix} \quad (11.12)$$

2. Write a MATLAB function that for a given vector x computes $f(x)$. Save as `f.m`. Test with $f(x) = e^x$ and $x = 0 : 3$.

```
>> f((0:3)')
```

ans =
1.0000
2.7183
7.3891
20.0855

3. As in Exercise 10.13, write a function `vdm.m` (for Vandermonde) that to a given vector $x = [x_1, \dots, x_m]^T \in \mathbb{R}^m$ and $p \in \mathbb{N}$ computes $V \in \mathbb{R}^{m \times p}$ with elements $v_{i,j} = x_i^{j-1}$, $i = 1, \dots, m$, $j = 1, \dots, p$. \triangleright MATLAB Hint¹¹ \triangleleft

```
>> vdm([1:2]', 3)
```

ans =
1 1 1
1 2 4

4. Write a function that given a vector x constructs the matrices A_1 and A_2 . Save as `eqos1.m`. Test with $x = (2 : 5)'$.

¹⁰ Use that $f(x_i) - (-1)^{i+1}f(x_1) = p(x_i) - (-1)^{i+1}p(x_1)$.

¹¹ Use `length` to obtain m from x .

```
>> [A1,A2]=eqos1 ((2:5) ')
A1 =
    1      3      9
    1      4     16
    1      5     25
A2 =
   -1     -2     -4
    1      2      4
   -1     -2     -4
```

5. Modify the inputs to write a function file with input the name of the file computing the function f and the vector \mathbf{x} and outputs \mathbf{b}_1 and \mathbf{b}_2 . Save as `eqos2.m`. Test with $x = (2 : 5)'$ and $namefile = 'f'$.

```
>> [b1,b2]=eqos2('f',(2:5)')
b1 =
    20.0855
    54.5982
    148.4132
b2 =
   -7.3891
    7.3891
   -7.3891
```

6. Complete the previous function with the solution of the linear system computing the coefficients of p . Save as `eqos3.m`. Same test.

```
>> coeff=eqos3('f',(2:5)')
coeff =
    127.1028
   -98.0732
    20.2796
```

7. Write a program that given the integer n and a function f defined on $[0, 1]$, computes uniform sites $\mathbf{x} = [0, \dots, x_i = i/n, \dots, 1]^T$ of $[0, 1]$, then computes the coefficients of the polynomial p such that $f - p$ is equioscillating, then plots the graphs of the polynomial p and the function f on $[0, 1]$. Test with $n = 5$, $f2(x) = |x - 1/3|$. Save as `grapheqos.m` (Fig. 11.4). Other test with $n = 15$.

Exercise 11.6. Remez algorithm

Evgeny Yakovlevich Remez (1896–1975) was a Soviet mathematician. He is known for his work in the constructive function theory, in particular, for the Remez algorithm and the Remez inequality.



Let $f \in C([0, 1])$. The goal of the exercise is to find a polynomial of degree at most $n - 1$, $p(t) = \sum_{k=1}^n \alpha_k t^{k-1}$ that minimizes $\|f - q\|_\infty = \max_{t \in [0, 1]} (|f(t) - q(t)|)$

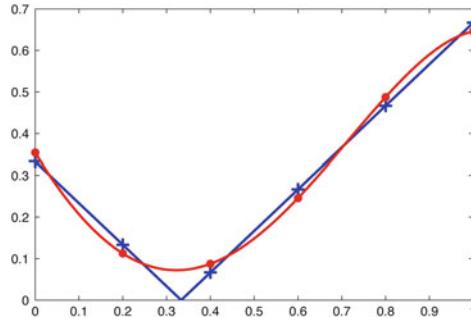


Fig. 11.4 Function $f(x) = |x - 1/3|$ and equioscillating polynomial p for $n = 5$

among all polynomial $q \in \mathbb{P}_{n-1}$. It can be proved that the solution p is such that $f - p$ is equioscillating in at least $n + 1$ distinct points x_1, \dots, x_{n+1} and,

$$\|f - p\|_\infty = \max_{0 \leq x_1 < \dots < x_{n+1} \leq 1} (|f(x_i) - p(x_i)|).$$

Exercise 11.5 gives the construction of the solution p .

The algorithm is iterative:

Initialisation: $n + 1$ points $0 \leq x_1^0 < x_2^0 < \dots < x_n^0 < x_{n+1}^0 \leq 1$ are chosen.

Step k: construct the polynomial $p \in \mathbb{P}_{n-1}$ such that

$$f(x_i^k) - p(x_i^k) = (-1)^{i+1}(f(x_1^k) - p(x_1^k)), \quad i = 2, \dots, n + 1.$$

1. If $\|f - p\|_\infty = \max_{i=1, \dots, n+1} (|f(x_i^k) - p(x_i^k)|)$, we are finished,

2. Else, there exists $y \in [0, 1]$ such that

$$\|f - p\|_\infty = |f(y) - p(y)| > \max_{i=1, \dots, n+1} (|f(x_i^k) - p(x_i^k)|).$$

Construct new sites $0 \leq x_1^{k+1} < \dots < x_{n+1}^{k+1} \leq 1$ by a modification of exactly one point x_i^k . There are 6 possibilities:

(a)	if $y \in [0, x_1^k]$ then $x_1^{k+1} = y$	and $(f(x_1^k) - p(x_1^k))(f(y) - p(y)) \geq 0$ and $x_i^{k+1} = x_i^k$ for $i = 2, \dots, n + 1$,
(b)	if $y \in [0, x_1^k]$ then $x_1^{k+1} = y$	and $(f(x_1^k) - p(x_1^k))(f(y) - p(y)) < 0$ and $x_i^{k+1} = x_{i-1}^k$ for $i = 2, \dots, n + 1$,
(c)	if $y \in (x_{n+1}^k, 1]$ then $x_{n+1}^{k+1} = y$	and $(f(x_{n+1}^k) - p(x_{n+1}^k))(f(y) - p(y)) \geq 0$ and $x_i^{k+1} = x_i^k$ for $i = 1, \dots, n$,
(d)	if $y \in (x_{n+1}^k, 1]$ then $x_{n+1}^{k+1} = y$	and $(f(x_{n+1}^k) - p(x_{n+1}^k))(f(y) - p(y)) < 0$ and $x_i^{k+1} = x_{i+1}^k$ for $i = 1, \dots, n$,
(e)	if $y \in (x_{i_0}^k, x_{i_0+1}^k)$ then $x_{i_0}^{k+1} = y$	and $(f(x_{i_0}^k) - p(x_{i_0}^k))(f(y) - p(y)) \geq 0$ and $x_i^{k+1} = x_i^k$ for $i = 1, \dots, n + 1, i \neq i_0$,
(f)	if $y \in (x_{i_0}^k, x_{i_0+1}^k)$ then $x_{i_0+1}^{k+1} = y$	and $x_i^{k+1} = x_i^k$ for $k = 1, \dots, n + 1, i \neq i_0 + 1$.

Programs:

First $\|f - p\|_\infty$ is approximated by $\max_{j=1,\dots,1001} |f(t_j) - p(t_j)|$ where $t = [t_1, \dots, t_{1001}]$ contains uniform sites on the interval, and the condition to obtain the optimal solution $\|f - p\|_\infty = \max_{i=1,\dots,n+1} (|f(x_i^k) - p(x_i^k)|)$ is replaced by $error < precis$ where $error = \max_{j=1,\dots,1001} |f(t_j) - p(t_j)| - \max_{i=1,\dots,n+1} (|f(x_i^k) - p(x_i^k)|)$ and $precis$ is a given (small) positive real. Then y is one of the t_j . Secondly the number of steps is limited by an integer $maxiter$.

1. Write a program that for a given vector x of ordered real numbers in $[0, 1]$, $0 \leq x_1 < x_2 \leq \dots \leq x_n \leq 1$ computes
 - a. The coefficients of the polynomial p using `eqos3.m`,
 - b. $e = \max_{i=1,\dots,n+1} (|f(x_i^k) - p(x_i^k)|)$,
 - c. The sites $t = \{t_j\}_{j=1}^{1001}$,
 - d. The $error = \max_{j=1,\dots,1001} |f(t_j) - p(t_j)| - e$,
 - e. The integer j_0 where the maximum is obtained. \triangleleft MATLAB Hint¹² \triangleright

Save as `errpol.m`. Test with $n = 10$, $x = 0 : 1/n : 1$, $f(x) = |x - 1/3|$. Show e , $error$ and j_0 . The vector $f(x)$ will be written fx and similar notations for $f(t)$, $p(x)$ and $p(t)$.

```
>> errpol
e =
    2.6632e-12
error =
    0.0576
j0 =
    34
```

2. Program Remez' Algorithm `Remezalgo.m`. \triangleleft MATLAB Hint¹³ \triangleright Plot the graphs of p and f . Test with $n = 10$, initialisation $x = 0 : 1/n : 1$, $f = f2$, $maxiter = 20$, $precis = 1e - 4$ (Fig. 11.5). \triangleleft MATLAB Hint¹⁴ \triangleright

11.2 Fourier Series Solution of the Heat Equation

Review: Let f be a piecewise C^1 function on $[-1, 1]$, i. e., with finite right and left limits at points where f and/or f' is discontinuous. Its Fourier series is given by

¹² Use `[m, j0] = max..`

¹³ Use `case` and `switch i`.

¹⁴ For $x_1 < x_2 < \dots < x_{n+1}$, the MATLAB instructions (`>>i=find(x>y,1)-1; if x(n+1)<y i=n+1; end`) gives an index $i = 0$ if $y < x_1$, $i = i_0$ if $x_{i_0} < y \leq x_{i_0+1}$ and $i = n + 1$ if $y > x_{n+1}$ (Fig. 11.5).

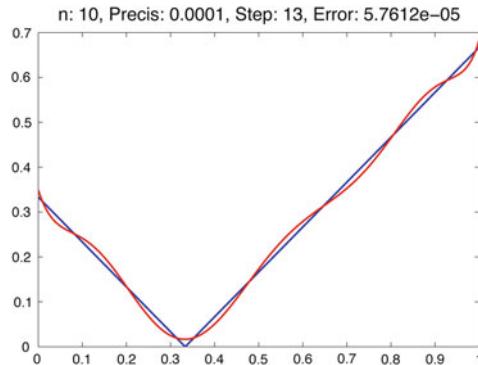


Fig. 11.5 Function $f(x) = |x - 1/3|$ and optimal polynomial p_n

$$S(x) := \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos(k\pi x) + b_k \sin(k\pi x),$$

where

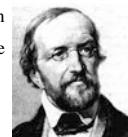
$$a_k = \int_{-1}^1 f(x) \cos(k\pi x) dx, \quad b_k = \int_{-1}^1 f(x) \sin(k\pi x) dx.$$

By Dirichlet's theorem

$$\lim_{n \rightarrow \infty} \left[\frac{a_0}{2} + \sum_{k=1}^n a_k \cos(k\pi x) + b_k \sin(k\pi x) \right] = \frac{f(x_-) + f(x_+)}{2}, \quad x \in [-1, 1],$$

where $f(x_+) := \lim_{t \rightarrow x, t > x} f(t)$ and $f(x_-) := \lim_{t \rightarrow x, t < x} f(t)$.

Johann Peter Gustav Lejeune Dirichlet (1805–1859) was a German mathematician who studied in Paris. He made contributions to number theory (Proof of Fermat's theorem for $n = 5$) and to the theory of Fourier series and other topics in mathematical analysis.



A function is **even** if $f(-x) = f(x)$ and **odd** if $f(-x) = -f(x)$ for all x . The Fourier series of an odd function reduces to a **Fourier sine series**

$$S(x) = \sum_{k=1}^{\infty} b_k \sin k\pi x, \quad b_k = 2 \int_0^1 f(x) \sin(k\pi x) dx.$$



Exercise 11.7. Heat equation

Consider a bar of length L in which heat is transmitted by conduction. It is assumed that the bar also has no heat exchange with the outside. By a change of variables, the simplified heat equation is given by

$$(P) \quad \begin{cases} \frac{\partial u}{\partial t}(x, t) - \frac{\partial^2 u}{\partial x^2}(x, t) = 0, & 0 \leq x \leq 1, 0 \leq t \\ u(x, 0) = u_0(x), & 0 \leq x \leq 1, \\ u(0, t) = u(1, t) = 0, & 0 \leq t, \end{cases} \quad \begin{array}{l} \text{initial conditions} \\ \text{end conditions.} \end{array} \quad (11.13)$$

For compatibility reasons, we assume that $u_0(0) = u_0(1) = 0$ and also that u_0 is a smooth enough function. In this case, (P) has a unique solution that can be written as

$$u(x, t) = \sum_{k=1}^{\infty} \gamma_k \sin(k\pi x) e^{-k^2\pi^2 t}, \quad x \in [0, 1], t > 0, \quad (11.14)$$

where

$$u_0(x) = \sum_{k=1}^{\infty} \gamma_k \sin(k\pi x), \quad \gamma_k = 2 \int_0^1 u_0(x) \sin(k\pi x) dx$$

is the Fourier sine series of the odd extension of u_0 to $[-1, 1]$.

In Chap. 13 we also compute approximations to the solution u of (P) using finite difference methods.

1. First example: Consider the function $u_0(x) = 1$ for $x \in (0, 1)$. Show that

$$\gamma_{2k} = 0, \quad \gamma_{2k-1} = \frac{4}{\pi} \frac{1}{2k-1}, \quad k = 1, 2, \dots$$

2. We can compute an approximation u_N to the initial function u_0 by taking a finite number of terms in the sine series for u_0

$$u_N(x) := \sum_{k=1}^N \gamma_k \sin(k\pi x). \quad (11.15)$$

Make MATLAB plots of the functions u_N in (11.15) for $N = 3, 10, 100$ for $u_0(x) = 1$. Save as `fourier1.m` (Fig. 11.6).

3. Second example: Consider the function $u_0(x) = x(1-x)$ for $x \in [0, 1]$. Find the γ_k s
4. Plot the approximate solution of (P) given by

$$u_n(x, t) = \sum_{k=1}^n \gamma_k \sin(k\pi x) e^{-k^2\pi^2 t}$$

when $u_0(x) = x(1-x)$ and $n = 10$ and $t \in [0, 0.4]$. See Fig. 11.7.

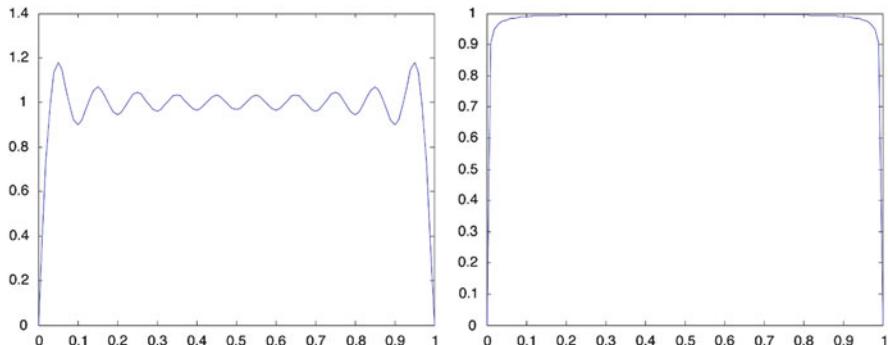


Fig. 11.6 The first N terms in a Fourier sine approximation to $u_0(x) = 1$. $N = 10$ left and $N = 100$ right

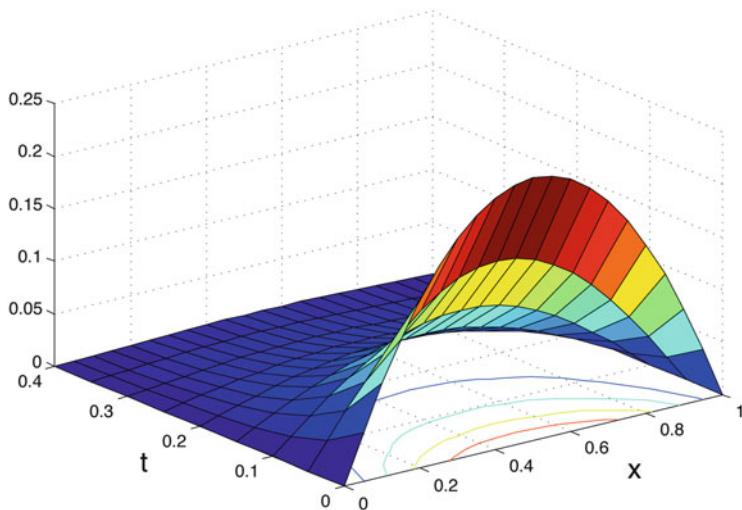


Fig. 11.7 The first 10 terms in the approximation of the solution of the heat equation

11.3 Discrete Signal and Haar Wavelets

Exercise 11.8. Haar wavelets

Alfred Haar (1885–1933) was a Hungarian mathematician. In 1904 he began to study at the University of Göttingen and his doctorate was supervised by David Hilbert. Together with Frigyes Riesz, he made the University of Szeged (Hungaria) a centre of mathematics. The Haar measure, Haar wavelet, and Haar transform are named in his honor.



We define the function $\phi : \mathbb{R} \rightarrow \mathbb{R}$ to be equal to one on the half open interval $[0, 1)$ and zero otherwise. We then define $\psi(x) := \phi(2x) - \phi(2x-1)$. Thus $\psi(x) = 1$ on $[0, 1/2)$ and $\psi(x) = -1$ on $[1/2, 1)$. From these functions we set

$$\varphi_{n,k}(x) := \varphi(2^n x - k) \text{ and } \psi_{n,k}(x) := \psi(2^n x - k), \quad n \in \mathbb{N}, \quad k = 0, \dots, 2^n - 1.$$

1. Plot the functions $\varphi, \psi, \varphi_{3,5}$ and $\psi_{3,5}$.
2. Prove that

$$\varphi_{n,k} = \varphi_{n+1,2k} + \varphi_{n+1,2k+1}, \quad \psi_{n,k} = \varphi_{n+1,2k} - \varphi_{n+1,2k+1}, \quad (11.16)$$

$$\varphi_{n+1,2k} = \frac{1}{2} [\varphi_{n,k} + \psi_{n,k}], \quad \varphi_{n+1,2k+1} = \frac{1}{2} [\varphi_{n,k} - \psi_{n,k}] \quad (11.17)$$

3. We recall that $L_2[0, 1]$ is the vector space of piecewise continuous functions f such that $\int_0^1 |f(t)|^2 dt$ is finite. On the vector space, there exists a scalar product: $\langle f, g \rangle = \int_0^1 f(t)g(t) dt$ with associated norm: $\|f\| = \langle f, f \rangle^{1/2}$.

We define $\tilde{\varphi}_{n,k} = 2^{n/2} \varphi_{n,k}$ and $\tilde{\psi}_{n,k} = 2^{n/2} \psi_{n,k}$.

Show that for $n \geq 0$ and $k, \ell = 0, \dots, 2^n - 1$,

$$\begin{aligned} 1 &= \|\tilde{\varphi}_{n,k}\| = \|\tilde{\psi}_{n,k}\|, \\ 0 &= \langle \varphi_{n,k}, \varphi_{n,\ell} \rangle = \langle \psi_{n,k}, \psi_{n,\ell} \rangle, \quad k \neq \ell \\ 0 &= \langle \varphi_{n,k}, \psi_{n,\ell} \rangle. \end{aligned}$$

4. Let V_n (respec. W_n) be the subspace of L_2 generated by the $\varphi_{n,k}$ (respec. $\psi_{n,k}$) for $k = 0, \dots, 2^n - 1$.

Prove that $\{\tilde{\varphi}_{n,k}\}_{k=0}^{2^n-1}$ ($\{\tilde{\psi}_{n,k}\}_{k=0}^{2^n-1}$) is an orthonormal basis for V_n (W_n) and show that $V_n \overset{\perp}{\oplus} W_n = V_{n+1}$ so that

$$\begin{aligned} V_{p-1} \overset{\perp}{\oplus} W_{p-1} &= V_p \subset L_2 \\ &\vdots \\ V_{n-1} \overset{\perp}{\oplus} W_{n-1} &= V_n \\ &\vdots \\ V_1 \overset{\perp}{\oplus} W_1 &= V_2 \\ V_0 \overset{\perp}{\oplus} W_0 &= V_1 \\ &V_0 \end{aligned}$$

5. Starting with $f_p \in V_p$, we define two finite sequences $f_n \in V_n$ and $g_n \in W_n$ with decreasing indices by $f_n = f_{n-1} + g_{n-1}$ for $n = p, \dots, 1$ i.e. f_{n-1} is an approximation of f_n in a smaller space and g_{n-1} gives the details to reconstruct f_n . We deduce that

$$f_p = f_0 + (g_0 + g_1 + \dots + g_n + \dots + g_{p-1}).$$

We assume that

$$f_n = \sum_{\ell=0}^{2^n-1} c_{n,\ell} \varphi_{n,\ell}, \quad n = 0, \dots, p, \quad g_n = \sum_{\ell=0}^{2^n-1} d_{n,\ell} \psi_{n,\ell}, \quad n = 0, \dots, p-1.$$

Compute the $c_{n-1,k}$ and $d_{n-1,k}$ depending on the $c_{n,\ell}$ (Decomposition) and conversely, compute $c_{n,2k}$ and $c_{n,2k+1}$ function of the $c_{n-1,\ell}$ and $d_{n-1,\ell}$ (Reconstruction).

Exercise 11.9. Compression-reconstruction

We start by a signal already studied in Exercise (10.13).

1. Write the function `signal1.m` that samples

$$\begin{aligned} F(x) &= \frac{1}{1 + 100(x - 0.2)^2} + \frac{1}{1 + 500(x - 0.4)^2} \\ &+ \frac{1}{1 + 1000(x - 0.6)^2} + \frac{1}{1 + 10000(x - 0.8)^2} \end{aligned}$$

at 2^p uniform points in $[0, 1 - 2^{-p}]$. The input is n and the outputs are x and y which will be written as rows of length 2^p . Test with $p = 2$.

```
>> [x,y]=signal1(2)
x =
      0      0.2500      0.5000      0.7500
y =
    0.2153      0.8901      0.3587      0.1291
```

2. Add a noise to the initial signal, $\mathbf{y}_n = \mathbf{y} + \frac{20}{100} \mathbf{v}$ where \mathbf{v} is a random vector with 2^p components between -1 and 1 . You can use the MATLAB function `rand.m` which gives random number in the interval $[0, 1]$. Save in `signal2.m`. Test with $p = 5$.

```
>> [x,y,z]=signal2(2)
x =
      0      0.2500      0.5000      0.7500
y =
    0.2153      0.8901      0.3587      0.1291
z =
    0.0661      0.9297      0.2115     -0.0248
```

Comment: The values of \mathbf{y}_n will normally be a little bit different since we have added random values. ☺

3. With the notations of Exercise 11.8, if $f_n = \sum_{k=0}^{2^n-1} y_k \varphi(2^n x - k) \in V_n$ where $\varphi(x) = \mathbb{1}_{[0,1]}(x)$ is the scale Haar function, the energy is defined by $e := \|f_n\|_{L_2} = \left(\frac{1}{2^n} \sum_{k=0}^{2^n-1} y_k^2 \right)^{1/2}$. Create a MATLAB function `energy.m` computing the energy of $\mathbf{y} = [y_0, \dots, y_{2^n-1}]^T$. Test

```
>> [x,y]=signal1(7); e=energy(y)
e =
    0.6244
```

4. Write a function `decomp.m` that given a signal $\mathbf{y} = [y_0, \dots, y_{2^p-1}]^T \in \mathbb{R}^{2^p}$ with the corresponding function $f_p = \sum_{k=0}^{2^p} y_k \varphi_{p,k} \in V_p$, computes the rows $[c_{n,0}, \dots, c_{n,2^n-1}]$ and $[d_{n,0}, \dots, d_{n,2^n-1}]$ in \mathbb{R}^{2^n} with the initialisation $c_{p,k} = y_k$ then the algorithm

Decomposition, $n = p$ downto 1:

$$\begin{cases} c_{n-1,k} &= \frac{c_{n,2k} + c_{n,2k+1}}{2} \\ d_{n-1,k} &= \frac{c_{n,2k} - c_{n,2k+1}}{2} \end{cases} \quad k = 0, \dots, 2^{n-1} - 1. \quad (11.18)$$

After calling `signal1`, the output of the program will be the matrix $\mathbf{D} \in \mathbb{R}^{p \times 2^{p-1}}$ and $c_{0,0} \in \mathbb{R}$.

$$\mathbf{D} = \begin{bmatrix} d_{p-1,0} & d_{p-1,1} & \dots & d_{p-1,2^{p-2}-1} & d_{p-1,2^{p-2}} & \dots & d_{p-1,2^{p-1}-1} \\ d_{p-2,0} & d_{p-2,1} & \dots & d_{p-2,2^{p-2}-1} & 0 & \dots & 0 \\ d_{p-3,0} & d_{p-3,1} & \dots & 0 & 0 & \dots & 0 \\ \vdots & & & & \vdots & & \vdots \\ d_{1,0} & d_{1,1} & 0 & \dots & 0 & \dots & 0 \\ d_{0,0} & 0 & 0 & \dots & 0 & \dots & 0 \end{bmatrix} \quad (11.19)$$

```
>> [x,y]=signal1(3);[D,c]=decomp(y)
D =
   -0.2276    -0.0690    -0.1752     0.0341
   -0.2581     0.2195      0          0
    0.1933      0          0          0
c =
    0.5077
```

5. **Compression:** Once we have computed $\mathbf{D} \in \mathbb{R}^{p \times 2^{p-1}}$ defined in (11.19) and $c_{0,0}$ with the decomposition algorithm, we compute $\tilde{\mathbf{D}}$ such that $\tilde{d}_{n,k} = 2^{-n/2} d_{n,k}$. Then, for a given $\epsilon > 0$, from \mathbf{D} , we consider the matrix $\bar{\mathbf{D}}$ and $\bar{c}_{0,0}$ such that

$$\bar{d}_{i,j} = \begin{cases} 0 & \text{if } |\tilde{d}_{i,j}| < \epsilon \\ d_{i,j} & \text{if } |\tilde{d}_{i,j}| \geq \epsilon \end{cases}, \quad \bar{c}_{0,0} = \begin{cases} 0 & \text{if } |c_{0,0}| < \epsilon \\ c_{0,0} & \text{if } |c_{0,0}| \geq \epsilon \end{cases}$$

The compression ratio is the number of non zero components of $\bar{\mathbf{D}}$ divided by the number of values of \mathbf{D} , i.e 2^p .

The MATLAB command `[r,c] = find(X)` returns the row and column indices of the nonzero entries in the matrix X , so that the command `[r,c] = find(abs(tD) >= epsi)` where $tD = \bar{D}$ will give the wanted indices.

Write a function `compress.m` searching for the indices of elements that are not too small. Inputs: a matrix A , and $epsi$, Output: indices r, c of the non zero entries of the matrix \bar{A} and $comp.ratio$. Test

```
>> [x,y]=signal1(3);[D,c]=decomp(y);
[r,c,compratio]=compress(D,0.1)
r =
    1
    2
    3
    2
c =
    1
    1
    1
    2
compratio =
    0.5000
```

6. Write the program `dcreconstruc.m` which computes the signal $x, y, D \in \mathbb{R}^{p \times 2^{p-1}}$ and $c_{0,0}$ defined in (11.19) and the vectors r, c with the compression program. The program will computed the reconstructed signal

$$y\ell(t) = \bar{c}_{0,0}\varphi_{0,0}(t) + \sum_{n=1}^p \sum_{k=0}^{2^n-1} \bar{d}_{n,k}\psi_{n,k}(t), \quad t \in [0, 1)$$

The sums will be computed only at the non vanishing $\bar{d}_{n,k}$. Also initial and reconstructed signals will be plotted and the compression ratio and energy ratio will be displayed. Test with $p = 8$ and $epsi = 0.01$;

```
>> dcreconstruc1
compressratio =
    0.1406
energyratio =
    0.9977
```

Graphs in Fig. 11.8

7. From `reconstruc1.m`, write the program `reconstruc2.m` which computes the signals x, y, yn with `signal2.m` and then applies the previous algorithm to yn instead of y . Test with $p = 10$ and $epsi = 0.02$

```
>> dcreconstruc2
compressratio =
    0.0205
energyratio =
    0.9785
```

Graphs in Fig. 11.9

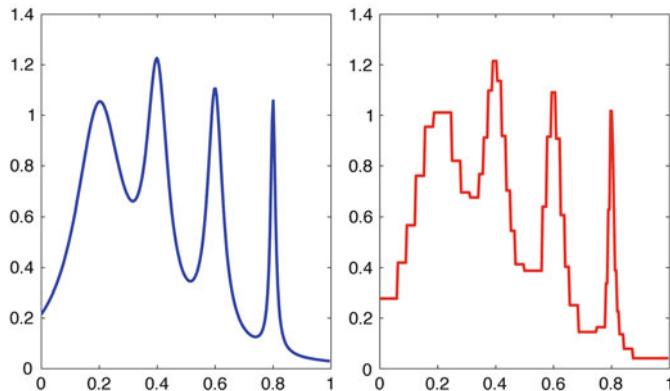


Fig. 11.8 Initial and Reconstructed Signals

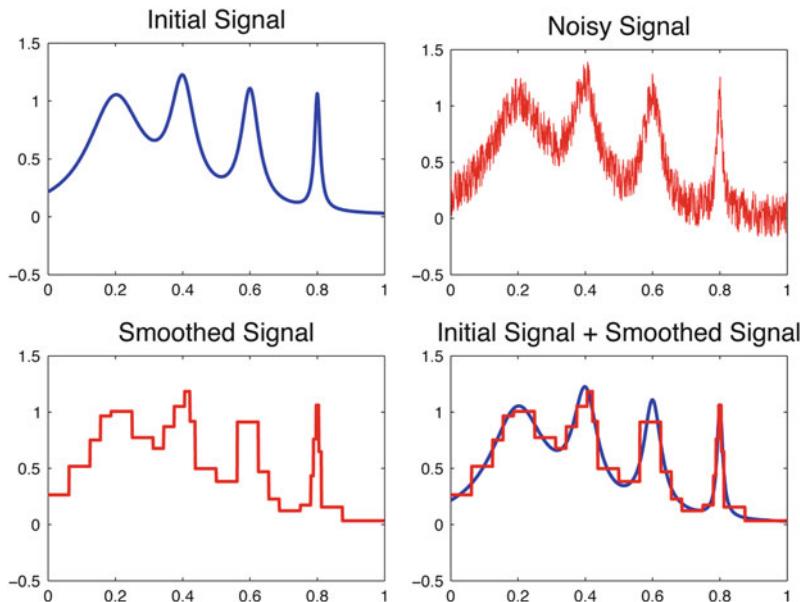


Fig. 11.9 Initial, Noisy and Reconstructed Signal

11.4 Solutions

11.4.1 Polynomial Approximation with $\|\cdot\|_\infty$

Solution of Exercise 11.1

1. $T_0 \in \mathbb{P}_0$ and $T_1 \in \mathbb{P}_1 \setminus \mathbb{P}_0$. If $T_p \in \mathbb{P}_p \setminus \mathbb{P}_{p-1}$ for $p = n$ and $p = n - 1$, then $2xT_n(x) - T_{n-1}(x)$ is a polynomial of exact degree $n + 1$ so that $T_{n+1} \in$

$\mathbb{P}_{n+1} \setminus \mathbb{P}_n$ which conclude the induction. Similarly the leading coefficient of T_n is 2^{n-1} .

2. Again we use induction to prove that for $x \in [-1, 1]$, $T_n(x) = \cos(n \arccos x)$. $T_0(x) = 1 = \cos(0 \arccos x)$ and $T_1(x) = x = \cos(1 \arccos x)$ for $x \in [-1, 1]$. Suppose $T_p(x) = \cos(p \arccos x)$ for $p = n - 1$ and $p = n$. Since $\cos(n+1)\theta + \cos(n-1)\theta = 2 \cos \theta \cos(n\theta)$, with $\theta = \arccos x$ for $x \in [-1, 1]$, we deduce that

$$\begin{aligned} T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \\ &= 2 \cos \theta \cos(n\theta) - \cos((n-1)\theta) = \cos((n+1)\theta). \end{aligned}$$

3. The solutions of $\cos(n \arccos x) = 0$ are given by $\arccos x = \frac{(2k-1)\pi}{2n}$ with $k \in \mathbb{Z}$. There are exactly n such distinct values in $[-1, 1]$ defined by $x_k = \cos\left(\frac{(2k-1)\pi}{2n}\right)$ for $k = 1, \dots, n$. Since $T_n \in \mathbb{P}_n$ has at most n roots, we have obtained all its roots.

From the previous questions it follows that $T_n(x) = 2^{n-1} \prod_{i=1}^n (x - x_i)$ and for $x \in [-1, 1]$, we know that $|T_n(x)| = |\cos(n \arccos x)|$ with maximum value 1, we find that $\|\prod_{i=1}^n (\cdot - x_i)\|_{\infty, [-1, 1]} = 2^{-(n-1)}$.

4. The extrema of T_n on $[-1, 1]$ are obtained by solving $\cos(n \arccos x) = \pm 1$ which gives $n + 1$ extrema: $x'_k = \cos\left(\frac{k\pi}{n}\right)$ for $k = 0, \dots, n$.
5. Chebyshevpol.m.

```
function T=chebyshevpol(n,x);
if n==0 T=ones(1,length(x));
else if n==1 T=x;
else R=chebyshevpol(n-1,x);
S=chebyshevpol(n-2,x);
T=2*x.*R-S;
end
end
```

6. plotchebypol.m

```
a=-1.2;b=1.2
x=a:(b-a)/500:b;
for n=0:8
y(n+1,:)=chebyshevpol(n,x);
end
for n=1:9
subplot(3,3,n)
axis([a b -2 2])
hold on
plot([a b],[-1,-1],'-r',[a b],[1,1],'-r',[ -1 -1],'\ldots
[-2,2],'-r',[1 1],[-2,2],'-r')
plot(x,y(n,:))
title(['n = ',int2str(n-1)]);
hold off
end
```

Solution of Exercise 11.2

1. Since $a \leq t \leq b$, we deduce that $-\frac{b-a}{2} \leq t - \frac{b+a}{2} \leq \frac{b-a}{2}$ and

$$-1 \leq x = \left(t - \frac{b+a}{2} \right) \left(\frac{2}{b-a} \right) \leq 1.$$

Similarly all the x_i are in $[-1, 1]$.

2. With the definition of x (resp. x_i), we obtain

$$f(t) - p_{n-1}(t) = \frac{f^{(n)}(\xi)}{n!} \prod_{j=1}^n \frac{b-a}{2} (x-x_j) = \frac{f^{(n)}(\xi)}{n!} \left(\frac{b-a}{2} \right)^n \prod_{j=1}^n (x-x_j)$$

3. If we choose the x_i to be the roots of the T_n defined in Exercise 11.1, we obtain

$$\left| \prod_{i=1}^n (x - x_i) \right| \leq 2^{-(n-1)} \text{ for } x \in [-1, 1].$$

4. Thus with $\|\cdot\|_\infty$ on $[a, b]$,

$$\|f - p_{n-1}\|_\infty \leq \frac{\|f^{(n)}\|_\infty}{n!} \left(\frac{b-a}{2} \right)^n 2^{-(n-1)} = \frac{2}{n!} \left(\frac{b-a}{4} \right)^n \|f^{(n)}\|_\infty.$$

We deduce the bound (11.2) for the minimum.

Solution of Exercise 11.3

1. For $x \in [0, 1]$, $x^i(1-x)^{n-i} \geq 0$, thus $B_i^n(x) \geq 0$.

Using the binomial formula followed by differentiation with respect to u , we obtain

$$(u+v)^n = \sum_{i=0}^n \binom{n}{i} u^i v^{n-i}, \quad (11.20)$$

$$n(u+v)^{n-1} = \sum_{i=0}^n \binom{n}{i} i u^{i-1} v^{n-i}, \quad (11.21)$$

$$n(n-1)(u+v)^{n-2} = \sum_{i=0}^n \binom{n}{i} i(i-1) u^{i-2} v^{n-i}. \quad (11.22)$$

In these expansions we set $u = x$, $v = 1 - x$, multiply (11.21) by x and (11.22) by x^2 . Then (11.5)–(11.7) follow.

Now

$$\begin{aligned}\sum_{i=0}^n (i-nx)^2 B_i^n(x) &= \sum_{i=0}^n [i(i-1) - (2nx-1)i + n^2 x^2] B_i^n(x) \\ &= r_2(x) - (2nx-1)r_1(x) + n^2 x^2 r_0(x) \\ &= n(n-1)x^2 - (2nx-1)nx + n^2 x^2 = nx(1-x)\end{aligned}$$

which is (11.8).

2. First, if $i \in I_x$ then $|f(i/n) - f(x)| \leq \varepsilon/2$ so that

$$\begin{aligned}\left| \sum_{i \in I_x} (f(i/n) - f(x)) B_i^n(x) \right| &\leq \sum_{i \in I_x} |f(i/n) - f(x)| B_i^n(x) \\ &\leq \varepsilon/2 \sum_{i \in I_x} B_i^n(x) \\ &\leq \varepsilon/2 \sum_{i=0}^n B_i^n(x) = \varepsilon/2.\end{aligned}$$

The last equality is obtained from (11.5).

Next, if $i \in J_x$, then $|i/n - x| > \eta$ so that $1 < \frac{(i-nx)^2}{n^2 \eta^2}$. We deduce that

$$\begin{aligned}\left| \sum_{i \in J_x} (f(i/n) - f(x)) B_i^n(x) \right| &\leq \sum_{i \in J_x} \frac{(i-nx)^2}{n^2 \eta^2} |f(i/n) - f(x)| B_i^n(x) \\ &\leq \frac{2M}{n^2 \eta^2} \sum_{i \in J_x} (i-nx)^2 B_i^n(x) \\ &\leq \frac{2M}{n^2 \eta^2} \sum_{i=0}^n (i-nx)^2 B_i^n(x) \\ &\stackrel{(11.8)}{=} \frac{2M}{n^2 \eta^2} nx(1-x) \leq \frac{M}{2\eta^2 n}\end{aligned}$$

since $x(1-x) \leq 1/4$.

3. Since $f(x) \stackrel{(11.5)}{=} \sum_{i=0}^n f(x) B_i^n(x)$ we obtain for $x \in [0, 1]$

$$\begin{aligned}|p_n(x) - f(x)| &= \left| \sum_{i=0}^n (f(i/n) - f(x)) B_i^n(x) \right| \\ &= \left| \sum_{i \in I_x} (f(i/n) - f(x)) B_i^n(x) + \sum_{i \in J_x} (f(i/n) - f(x)) B_i^n(x) \right| \\ &\leq \varepsilon/2 + \frac{M}{2\eta^2 n}.\end{aligned}$$

Now $\frac{M}{2\eta^2 n} \rightarrow 0$ as $n \rightarrow +\infty$, so that $0 \leq \frac{M}{2\eta^2 n} \leq \varepsilon/2$ for n large enough, independently of x . We deduce that $\|f - p_n\|_\infty < \varepsilon$ for n large enough.

Solution of Exercise 11.4

1. decasteljau.m

```
function p=decasteljau(y, t)
N=length(y); %N=n+1
%initialization
[B, TT]=ndgrid(y, t);
%de Casteljau
for r=N-1:-1:1
    B(1:r,:)=(1-TT(1:r,:)).*B(1:r,:)+TT(1:r,:).*B(2:r+1,:);
end
```

2. errorstudy1.m

```
arrn=100:110;
namefunction='f1';
t=0:1/1000:1;
yex=feval(namefunction , t );
for j=1:length(arrn)
    n=arrn(j);
    x=0:1/n:1;
    y=feval(namefunction ,x );
    p=decasteljau(y, t);
    arrerror(j)=norm(p-yex , inf );
end
plot(log(arrn),log(arrerror) , 'o—' , 'Linewidth' ,2)
a=polyfit(log(arrn),log(arrerror) ,1);
xlabel('log(n)' , 'Fontsize' ,14)
ylabel('log(error)' , 'Fontsize' ,14)
title(['Error ||f-p_n|| . Slope : ',num2str(a(1))] , 'Fontsize' ,18)
```

3. If $f(t) = 1$ for all t then $p_n(t) = \sum_{i=0}^n B_i^n(t) \stackrel{(11.5)}{=} 1 = f(t)$. Similarly, if $f(t) = t$ for all t then $p_n(t) = \sum_{i=0}^n \frac{i}{n} B_i^n(t) \stackrel{(11.6)}{=} t = f(t)$. Finally, if $f(t) = a + bt$ for all t then

$$f(t) = a \sum_{i=0}^n B_i^n(t) + b \sum_{i=0}^n \frac{i}{n} B_i^n(t) = \sum_{i=0}^n f\left(\frac{i}{n}\right) B_i^n(t) = p_n(t).$$

4. If $f \in \mathbb{P}_1$, with the previous result, we obtain that for any $\alpha_n \in \mathbb{R}$, $f = \alpha_n p_{2n} + (1 - \alpha_n)p_n$.

Let $f(t) = t^2$, then for $n \geq 2$ by (11.6) and (11.7)

$$p_n(t) = \sum_{i=0}^n \left(\frac{i}{n}\right)^2 B_i^n(t) = \sum_{i=0}^n \frac{i(i-1)}{n^2} B_i^n(t) + \sum_{i=0}^n \frac{i}{n^2} B_i^n(t) = \frac{n-1}{n} t^2 + \frac{1}{n} t, \quad (11.23)$$

from which we deduce

$$\begin{aligned}
\alpha_n p_{2n}(t) &+ (1 - \alpha_n) p_n(t) \\
&= \alpha_n \left(\frac{2n-1}{2n} t^2 + \frac{1}{2n} t \right) + (1 - \alpha_n) \left(\frac{n-1}{n} t^2 + \frac{1}{n} t \right) \\
&= \left[\alpha_n \left(1 - \frac{1}{2n} \right) + (1 - \alpha_n) \left(1 - \frac{1}{n} \right) \right] t^2 \\
&\quad + \frac{1}{n} \left[\frac{\alpha_n}{2} + (1 - \alpha_n) \right] t
\end{aligned}$$

So that $\alpha_n p_{2n}(t) + (1 - \alpha_n) p_n(t) = t^2$ as soon as $\alpha_n = 2$.

With the linearity, we conclude that $f = q_n$ for $n \geq 2$ and any $f \in \mathbb{P}_2$.

5. If $f \in \mathbb{P}_1$, then $p_n = f$ and $p_n'' = 0$ so that for any $\bar{\alpha}_n$, $p_n - \bar{\alpha}_n p_n'' = p_n = f$.

From $p_n(t) = \sum_{i=0}^n f\left(\frac{i}{n}\right) B_i^n(t)$, with (7.3), we obtain that

$p_n''(t) = n(n-1) \sum_{i=0}^{n-2} \Delta^2 f\left(\frac{i}{n}\right) B_i^{n-2}(t)$, and if $f(t) = t^2$, then

$$\Delta^2 f\left(\frac{i}{n}\right)^2 = \frac{(i+2)^2 - 2(i+1)^2 + i^2}{n^2} = \frac{2}{n^2} \text{ with (11.5) so that } p_n''(t) = \frac{2(n-1)}{n}.$$

Using (11.23), we obtain

$$r_n(t) = p_n(t) - \bar{\alpha}_n(t) p_n''(t) = \frac{n-1}{n} t^2 + \frac{1}{n} t - \bar{\alpha}_n(t) \frac{2(n-1)}{n} \text{ and } r_n(t) = t^2 \text{ for } \bar{\alpha}_n(t) = \frac{t(1-t)}{2(n-1)}.$$

Thus, with the linearity, the result is proved for any $f \in \mathbb{P}_2$ and $\bar{\alpha}_n(t) = \frac{t(1-t)}{2(n-1)}$.

6. `errorstudy2.m`

```

arrn=100:105;
namefunction='f';
t=0:1/1000:1;
yex=feval(namefunction , t );
for j=1:length(arrn)
    n=arrn(j);
    x=0:1/n:1;
    y1=feval(namefunction , x );
    p1=decasteljau(y1 , t );
    delta2y1=y1(3:n+1)-2*y1(2:n)+y1(1:n-1);
    s_1=decasteljau(n*(n-1)*delta2y1 , t );
    r=p1-t.*((1-t)/(2*(n-1)).*s_1 ;
    x=0:1/(2*n):1;
    y2=feval(namefunction , x );
    p2=decasteljau(y2 , t );
    q=2*p2-p1;

    arrrror1(j)=norm(q-yex , inf );
    arrrror2(j)=norm(r-yex , inf );
end
subplot(1,2,1)
plot(log(arrn),log(arrrror1),'+-','Linewidth',2)
a=polyfit(log(arrn),log(arrrror1),1);
xlabel('log(n)', 'FontSize',14)

```

```

ylabel('log(error)', 'FontSize', 14)
title(['Error ||f-q_n||. Slope: ', num2str(a(1))], 'FontSize', 18)
subplot(1,2,2)
plot(log(arrn), log(arrerror2), 'o—', 'LineWidth', 2)
a=polyfit(log(arrn), log(arrerror2), 1);
title(['Error ||f-r_n||. Slope: ', num2str(a(1))], 'FontSize', 18)
xlabel('log(n)', 'FontSize', 14)
ylabel('log(error)', 'FontSize', 14)

```

In both cases, the order of convergence is 2.

Solution of Exercise 11.5

1. With $p(t) = \sum_{k=1}^n \alpha_k t^{k-1}$, (11.10) becomes

$$\begin{aligned}
& f(x_i) - (-1)^{i+1} f(x_1), \quad i = 2, \dots, n+1 \\
&= p(x_i) - (-1)^{i+1} p(x_1) = \sum_{k=1}^n \alpha_k x_i^{k-1} - (-1)^{i+1} \sum_{k=1}^n \alpha_k x_1^{k-1} \\
&= \left([1 \quad x_i \quad x_i^2 \quad \dots \quad x_i^{n-1}] - (-1)^{i+1} [1 \quad x_1 \quad x_1^2 \quad \dots \quad x_1^{n-1}] \right) \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_n \end{bmatrix}
\end{aligned}$$

which gives $A\alpha = b$ where $A = A_1 - A_2 \in \mathbb{R}^{n \times n}$ and $b = b_1 - b_2 \in \mathbb{R}^n$ defined by (11.11) and (11.12).

2. f.m

```

function y=f(x)
y=exp(x);

```

3. vdm.m

```

function A=vdm(x, n)
m=length(x);
A=ones(m, n);
for j=1:n-1
    A(:, j+1)=A(:, j).*x;
end;

```

4. eqos1.m.

```

function [A1,A2]=eqos1(x)
n=length(x)-1;
A1=vdm(x(2:n+1),n);
A2=cos(pi*(1:n)').*vdm(x(1),n);

```

5. eqos2.m

```
function [b1,b2]=eqos2(namefunc,x)
n=length(x)-1;
b1=feval(namefunc,x(2:n+1));
b2=feval(namefunc,x(1))*cos(pi*(1:n));
```

6. eqos3.m

```
function coeff=eqos3(namefunc,x)
n=length(x)-1;
A=vdm(x(2:n+1),n)-cos(pi*(1:n'))*vdm(x(1),n);
b=feval(namefunc,x(2:n+1)) - feval(namefunc,x(1))*cos(pi*(1:n'));
coeff=A\b;
```

7. grappheqos.m

```
n=5;
namefunct='f2';
x=(0:1/n:1)';
coeff=eqos3(namefunct,x);
t=0:1/500:1;
yf=feval(namefunct,t);
yp=polyval(coeff(n:-1:1),t);
plot(x,feval(namefunct,x),'+b',t,yf,'b',...
x,polyval(coeff(n:-1:1),x),'or',t,yp,'r')
```

Solution of Exercise 11.6

1. errpol.m

```
n=10;
namefunc='f';
x=0:1/n:1;
coeff=eqos3(namefunc,x');
fx=feval(namefunc,x');
px=polyval(coeff(n:-1:1),x');
e=norm(fx-px,inf);
t=0:1/1000:1;
ft=feval(namefunc,t');
pt=polyval(coeff(n:-1:1),t');
[error,j0]=max(abs(ft-pt)-e);
plot(t,pt,t,ft)
```

2. Remezalgo.m

```
%initialisation: computation of x,t,px,pt,fx,ft... e and error
close all
n=10;
maxiter=20;
precis=1e-4;
namefunc='f';
x=0:1/n:1;
```

```

coeff=eqos3(namefunc,x');
fx=feval(namefunc,x');
px=polyval(coeff(n:-1:1),x');
e=norm(fx-px,inf);
t=0:1/1000:1;
ft=feval(namefunc,t');
pt=polyval(coeff(n:-1:1),t');
[error,j0]=max(abs(ft-pt)-e);

%iterations
iter=0;
while (error>precis)&(iter<= maxiter)
    y=t(j0);
    i=find(x>y,1)-1; if x(n+1)<y i=n+1; end
    switch i
        case 0
            if (fx(1)-px(1))*(ft(j0)-pt(j0))>=0 x(1)=y;
            else x(2:n+1)=x(1:n); x(1)=y;
            end
        case n+1
            if (fx(n+1)-px(n+1))*(ft(j0)-pt(j0))>=0 x(n+1)=y;
            else x(1:n)=x(2:n+1); x(n+1)=y;
            end
        otherwise
            if (fx(i)-px(i))*(ft(j0)-pt(j0))>=0 x(i)=y;
            else x(i+1)=y;
            end
    end
    coeff=eqos3('f',x');
    fx=feval(namefunc,x');
    px=polyval(coeff(n:-1:1),x');
    e=norm(fx-px,inf);
    t=0:1/1000:1;
    ft=feval(namefunc,t');
    pt=polyval(coeff(n:-1:1),t');
    [error,j0]=max(abs(ft-pt)-e)
    iter=iter+1
end

plot(t,ft,t,pt)
error
title(['n: ',int2str(n),', Precis: ',num2str(precis),...
        ', Step: ',int2str(iter-1),', Error: ',num2str(error)])
```

,

11.4.2 Fourier Series Solution of the Heat Equation

Solution of Exercise 11.7

1. First example

$$\gamma_{2k} = 2 \int_0^1 \sin(2k\pi x) dx = -\frac{2}{(2k-1)\pi} [\cos(2k\pi x)]_0^1 = 0,$$

$$\gamma_{2k-1} = 2 \int_0^1 \sin((2k-1)\pi x) dx = -\frac{2}{(2k-1)\pi} [\cos((2k-1)\pi x)]_0^1 = \frac{4}{\pi} \frac{1}{2k-1}.$$

2. fourier1.m

```
N=3;
M=200;
x=0:1/M:1;
V=zeros(N,M+1);
for k=1:N
    V(k,:)=4/( pi*(2*k-1))*sin((2*k-1)*pi*x);
end
plot(x,sum(V))
```

3. We find

$$\begin{aligned}\frac{1}{2}\gamma_k &:= \int_0^1 x(1-x)\sin(k\pi x) dx \\ &= \left[-x(1-x)\frac{\cos(k\pi x)}{k\pi} \right]_0^1 + \int_0^1 (1-2x)\frac{\cos(k\pi x)}{k\pi} dx \\ &= \left[(1-2x)\frac{\sin(k\pi x)}{k^2\pi^2} \right]_0^1 + 2 \int_0^1 \frac{\sin(k\pi x)}{k^2\pi^2} dx \\ &= -\left[2\frac{\cos(k\pi x)}{k^3\pi^3} \right]_0^1 = \frac{2}{k^3\pi^3}(1 - (-1)^k).\end{aligned}$$

which gives

$$\gamma_{2k} = 0, \quad \gamma_{2k-1} = \frac{8}{\pi^3} \frac{1}{(2k-1)^3}, \quad k = 1, 2, \dots$$

4. heat1.m

```
N=10;
M=20;
x=0:1/M:1;
t=0:1/M:0.4;
[xx, tt ]=meshgrid(x, t);
V=zeros(length(t), length(x));
for k=1:N
    V=V+8/( pi*(2*k-1))^3* sin((2*k-1)*pi*xx).* exp(-k^2*pi^2*t);
end
surf(xx, tt ,V)
xlabel('x')
ylabel('t')
```

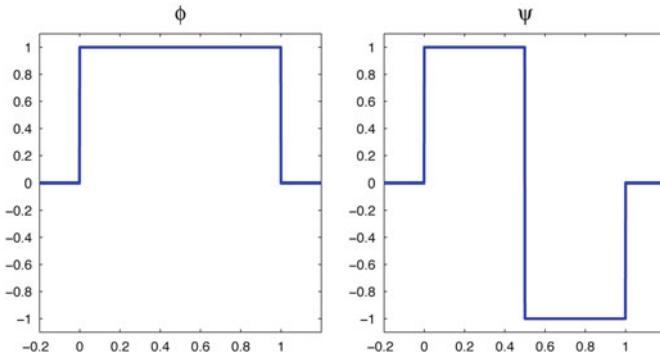


Fig. 11.10 Functions φ and ψ

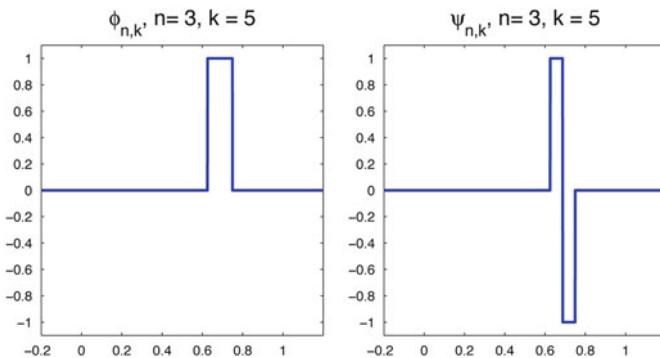


Fig. 11.11 Functions $\varphi_{3,5}$ and $\psi_{3,5}$.

11.4.3 Discrete Signal and Haar Wavelets

Solution of Exercise 11.8

- Graphs of φ and ψ are in Fig. 11.10, graphs of $\varphi_{3,5}$ and $\psi_{3,5}$ in Fig. 11.11.
- We have $\varphi(2X) = 0$ or 1 and $\varphi(2X - 1) = 0$ or 1 for $X \in \mathbb{R}$. Testing successively for $X < 0$, $0 \leq X < 1/2$, $1/2 \leq X < 1$ and $1 \leq X$, we obtain $\varphi(X) = \varphi(2X) + \varphi(2X - 1)$ and by its definition, $\psi(X) := \varphi(2X) - \varphi(2X - 1)$. Now if $x \in \mathbb{R}$, with $X = 2^n x - k$ in the previous equalities, we obtain (11.16) from which we deduce (11.17) by sum and difference.

3. For $n \geq 0$, $\tilde{\varphi}_{n,k}(x) = \begin{cases} 2^{n/2} & \text{if } x \in [k2^{-n}, (k+1)2^{-n}) \\ 0 & \text{if } x \notin [k2^{-n}, (k+1)2^{-n}), \end{cases}$ $k = 0, \dots, 2^n - 1$, so that

$$\int_0^1 |\tilde{\varphi}_{n,k}(x)|^2 dt = \int_{k2^{-n}}^{(k+1)2^{-n}} |\tilde{\varphi}_{n,k}(x)|^2 dt = 2^n \times 2^{-n} = 1,$$

thus $1 = \|\tilde{\varphi}_{n,k}\|$.

Since

$$\tilde{\psi}_{n,k}(x) = \begin{cases} 2^{n/2} & \text{if } x \in [k2^{-n}, (k+1/2)2^{-n}) \\ -2^{n/2} & \text{if } x \in [(k+1/2)2^{-n}, (k+1)2^{-n}) \\ 0 & \text{if } x \notin [k2^{-n}, (k+1)2^{-n}) \end{cases},$$

we obtain similarly $1 = \|\tilde{\psi}_{n,k}\|$.

For the second part, the support of $\varphi_{n,k}$ or $\psi_{n,k}$ is $[k2^{-n}, (k+1)2^{-n}]$. We deduce that for $k \neq \ell$ the support of $\varphi_{n,k}$ and $\varphi_{n,\ell}$ are disjoint except perhaps one point, precisely $(k+1)2^{-n}$ if $k+1 = \ell$ and $k2^{-n}$ if $k-1 = \ell$. Thus

$$\langle \varphi_{n,k}, \varphi_{n,\ell} \rangle = \int_0^1 \varphi_{n,k}(t) \varphi_{n,\ell}(t) dt = 0, \quad k \neq \ell.$$

Similarly $\langle \psi_{n,k}, \psi_{n,\ell} \rangle = 0 = \langle \varphi_{n,k}, \psi_{n,\ell} \rangle$ for $k \neq \ell$.

Finally $\langle \varphi, \psi \rangle = \int_0^1 \varphi(x) \psi(x) dx = 0$ since on the interval $[0, 1]$, the functions $\varphi(x) = 1$ and $\psi(x) = \begin{cases} 1, & x \in [0, 1/2) \\ -1, & x \in [1/2, 1] \end{cases}$. Then with $x = 2^n t - k$, we obtain

$$\begin{aligned} \langle \varphi_{n,k}, \psi_{n,k} \rangle &= \int_0^1 \varphi_{n,k}(t) \psi_{n,k}(t) dt = \int_{k2^{-n}}^{(k+1)2^{-n}} \varphi_{n,k}(t) \psi_{n,k}(t) dt \\ &= \int_{k2^{-n}}^{(k+1)2^{-n}} \varphi(2^n t - k) \psi(2^n t - k) dt \\ &= 2^{-n} \int_0^1 \varphi(x) \psi(x) dx = 0. \end{aligned}$$

4. Since $1 = \|\tilde{\varphi}_{n,k}\|$ and for $k \neq \ell$, $\langle \tilde{\varphi}_{n,k}, \tilde{\varphi}_{n,\ell} \rangle = 2^n \langle \varphi_{n,k}, \varphi_{n,\ell} \rangle = 0$, we deduce that the set $\{\tilde{\varphi}_{n,k}\}_{k=0, \dots, 2^n-1}$ is an orthonormal basis for V_n and similar arguments prove that $\{\tilde{\psi}_{n,k}\}_{k=0, \dots, 2^n-1}$ is an orthonormal basis for W_n . Since $\langle \varphi_{n,k}, \psi_{n,\ell} \rangle = 0$, we see that $V_n \overset{\perp}{\oplus} W_n$ is an orthogonal sum.

Now, using the basis for each subspace, (11.16) gives $V_n \subset V_{n+1}$ and $W_n \subset V_{n+1}$ while (11.17) proves that $V_{n+1} \subset V_n + W_n$.

Finally we see that $V_{n+1} = V_n \overset{\perp}{\oplus} W_n$.

5. For $n = 1, \dots, p$, with (11.17) applied to $n-1$ instead of n and ℓ instead of k , we obtain

$$\begin{aligned}
f_n &= \sum_{\ell=0}^{2^n-1} c_{n,\ell} \varphi_{n,\ell} = \sum_{k=0}^{2^{n-1}-1} c_{n,2k} \varphi_{n,2k} + \sum_{k=0}^{2^{n-1}-1} c_{n,2k+1} \varphi_{n,2k+1} \\
&= \sum_{k=0}^{2^{n-1}-1} c_{n,2k} \frac{1}{2} [\varphi_{n-1,k} + \psi_{n-1,k}] + \sum_{k=0}^{2^{n-1}-1} c_{n,2k+1} \frac{1}{2} [\varphi_{n-1,k} - \psi_{n-1,k}] \\
&= \sum_{k=0}^{2^{n-1}-1} \frac{c_{n,2k} + c_{n,2k+1}}{2} \varphi_{n-1,k} + \sum_{k=0}^{2^{n-1}-1} \frac{c_{n,2k} - c_{n,2k+1}}{2} \psi_{n-1,k} \\
&= f_{n-1} + g_{n-1} = \sum_{k=0}^{2^{n-1}-1} c_{n-1,k} \varphi_{n-1,k} + \sum_{k=0}^{2^{n-1}-1} d_{n-1,k} \psi_{n-1,k}.
\end{aligned}$$

We deduce that

$$\text{Decomposition: } \begin{cases} c_{n-1,k} &= \frac{c_{n,2k} + c_{n,2k+1}}{2} \\ d_{n-1,k} &= \frac{c_{n,2k} - c_{n,2k+1}}{2} \end{cases} \quad k = 0, \dots, 2^{n-1} - 1.$$

While computing the sum and difference of the two previous equations, we deduce that

$$\text{Reconstruction: } \begin{cases} c_{n,2k} &= c_{n-1,k} + d_{n-1,k} \\ c_{n,2k+1} &= c_{n-1,k} - d_{n-1,k} \end{cases} \quad k = 0, \dots, 2^{n-1} - 1.$$

Solution of Exercise 11.9

1. signal1.m

```

function [x,y]=signal1(p);
P=2^p;
h=1/P;
x=[0:h:1-h];
y=1./(1+100*(x-0.2).^2)+1./(1+500*(x-0.4).^2)+...
    1./(1+1000*(x-0.6).^2)+1./(1+10000*(x-0.8).^2);

```

2. signal2.m

```

function [x,y,yn]=signal2(p);
P=2^p;
h=1/P;
x=[0:h:1-h];
y=1./(1+100*(x-0.2).^2)+1./(1+500*(x-0.4).^2)+...
    1./(1+1000*(x-0.6).^2)+1./(1+10000*(x-0.8).^2);
V=2*rand(1,P)-1;
yn=y+V*0.2;

```

3. energy.m

```

function e=energy(y)
P=length(y);
e=norm(y,2)/sqrt(P);

```

4. decomp.m

```
function [D,C]=decomp(y)
p=round(log(length(y))/log(2));
P=2^p;
C=y;
Q=P;
for i=1:p
    C1=C(1:2:Q-1);
    C2=C(2:2:Q);
    Q=Q/2;
    C=(C1+C2)/2;
    D(i,1:Q)=(C1-C2)/2;
end
```

5. compress.m

```
function [r,c,compratio]=decomp(A,epsi)
p=length(A(:,1));
tA=A.*((2.^(-[p-1:-1:0]/2))'*ones(1,2^(p-1)));
[r,c]=find(abs(tA)>epsi);
compratio=length(r)/(2^p);
```

6. dcreconstruc1.m

```
p=8;[x,y]=signal1(p);
e1=energy(y);
epsi=0.01;
[D,c00]=decomp(y);
[r,c,compressratio]=compress(D,epsi);
yl=c00*find(abs(c00)>epsi)*fphi(x);
for i=1:length(r)
    yl=yl+D(r(i),c(i))*fpsi(2^(p-r(i))*x-c(i)+1);
end
e2=energy(yl)
compressratio
energyratio=e2/e1
subplot(1,2,1)
plot(x,y,'b')
subplot(1,2,2)
plot(x,yl,'r')
```

dcreconstruc2.m

```
p=10;[x,y,yn]=signal2(p);
e1=energy(yn);
epsi=0.02;
[D,c00]=decomp(yn);
[r,c,compressratio]=compress(D,epsi);
yl=c00*find(abs(c00)>epsi)*fphi(x);
for i=1:length(r)
    yl=yl+D(r(i),c(i))*fpsi(2^(p-r(i))*x-c(i)+1);
end
e2=energy(yl);
compressratio
```

```
energyratio=e2/e1
subplot(2,2,1)
plot(x,y,'b')
title('Initial Signal')
axis([0 1 -0.5 1.5])
subplot(2,2,2)
plot(x,yn,'r')
title('Noisy Signal')
axis([0 1 -0.5 1.5])
subplot(2,2,3)
plot(x,yl,'r')
axis([0 1 -0.5 1.5])
title('Smoothed Signal')
subplot(2,2,4)
plot(x,y,'b',x,yl,'r')
axis([0 1 -0.5 1.5])
title('Initial Signal + Smoothed Signal')
```

Chapter 12

Ordinary Differential Equations, One Step Methods

In this chapter we study initial value problems for systems of ordinary differential equations. We also consider some boundary value problems. Such differential systems occur often in science and engineering. Existence and uniqueness of a solution can be proved under general conditions, but it is in general impossible to write down the exact solution. Numerical methods (ODE solvers) provide tools to obtain discrete approximations. We give examples of differential systems and some numerical methods for their approximation.

Three problems are considered in Sect. 12.1. We test the Euler and backward Euler methods in Exercise 12.1. In Exercise 12.2, we compute the exact solution of a differential system of order one with initial conditions, then in Exercise 12.3, we approximate the exact solution with the two Euler methods.

Then, in Sect. 12.2, we consider two Runge-Kutta methods. They are known as **Heun's method** and **Fourth order Runge-Kutta**. We consider Heun's method in Exercise 12.4 with emphasis on theory, stability and order of convergence. Then in Exercises 12.5 and 12.6 we apply Heun's method to two problems; the problem solved in Exercise 12.1 and a nonlinear equation simulating the movements of a spring. In a second part of this section, we solve two problems using the fourth order Runge-Kutta method. We consider a problem with fourth derivatives in Exercises 12.7 and a first order equation in 12.8, where the numerical method has problems extending the solution. We include numerical studies of the errors to confirm the order of the methods.

Section 12.3 is devoted to a **predictor-corrector method** with an example in Exercise 12.9. We remark that any of the other differential problems in this chapter can be studied by these methods.

Finally, in Sect. 12.4, we show how some boundary value problems can be transformed into initial value problems so that ODE solvers can be used. The method is called the **shooting method** and is proposed in Exercises 12.10, 12.11 and 12.12. A third order Runge-Kutta method is used in Exercise 12.11.

Many ODE solvers are implemented in MATLAB. Examples are `ode23`, `ode45`, `ode113`. The output is a discrete solution \mathbf{u} at points $t = [t_1, \dots, t_{n+1}]$ using variable step length.

Review: For $k, m \in \mathbb{N}$, $r \in \mathbb{Z}$, $r \geq 0$, $X \subset \mathbb{R}^k$ and $Y \subset \mathbb{R}^m$ we say that a function $f : X \rightarrow Y$ belongs to the class $C^r(X, Y)$ if all partial derivatives of f of order $\leq r$ are continuous on X .

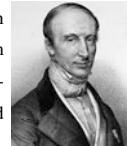
For $I = [t_1, t_1 + T] \subset \mathbb{R}$, let $f(t, \mathbf{x}) \in C(I \times \mathbb{R}^m, \mathbb{R}^m)$. With an initial condition $\boldsymbol{\eta} \in \mathbb{R}^m$, we are looking for a function $\mathbf{y} \in C^1(I, \mathbb{R}^m)$ solving the **Cauchy problem**

$$\mathbf{y}'(t) = f(t, \mathbf{y}(t)), \forall t \in I, \quad (12.1)$$

$$\mathbf{y}(t_1) = \boldsymbol{\eta}. \quad (12.2)$$

This is an initial value problem for a system of first order differential equations and (12.2) is called the **initial condition**.

Baron **Augustin-Louis Cauchy** (1789–1857) was a French mathematician. He defined continuity in terms of infinitesimals and gave theorems in complex analysis. He started as an engineer and from 1815, he taught mathematics at the École Polytechnique. After an exile (Turin, Berlin), Cauchy returned to Paris and his position at the Academy of Sciences. He wrote approximately eight hundred research articles and five complete textbooks.

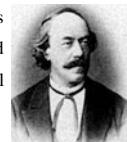


Suppose $f(t, \mathbf{x}) \in C(I \times \mathbb{R}^m, \mathbb{R}^m)$. Given a norm $\|\cdot\|$ on \mathbb{R}^m , if f is continuous and satisfies a **Lipschitz condition** in its second variable i.e. there exists a positive real number L such that

$$\forall t \in I, \forall \mathbf{y}, \mathbf{z} \in \mathbb{R}^m, \|f(t, \mathbf{y}) - f(t, \mathbf{z})\| \leq L \|\mathbf{y} - \mathbf{z}\| \quad (12.3)$$

then the problem (12.1)–(12.2) has a unique solution.

Rudolf Otto Sigismund Lipschitz (1832–1903) was a German mathematician. While he gave his name to the Lipschitz continuity condition, he worked in a broad range of areas. These included number theory, algebras with involution, mathematical analysis, differential geometry and classical mechanics.



Numerical schemes: We compute approximations $\mathbf{u}_i \approx \mathbf{y}(t_i)$ on a **partition** $t : t_1 < t_2 < \dots < t_{n+1} = t_1 + T$ of the interval $I = [t_1, t_1 + T]$. Here $\mathbf{u}_1 \in \mathbb{R}^m$ is given and

$$\mathbf{u}_{i+1} := \mathbf{u}_i + h_i \phi(t_i, \mathbf{u}_i, h_i),, \quad h_i := t_{i+1} - t_i, \quad i = 1, \dots, n. \quad (12.4)$$

The function $\phi : I \times \mathbb{R}^m \times R \rightarrow \mathbb{R}^m$, is a continuous function defining the numerical method used. We are interested in methods that give accurate approximations when $H := \max h_i$ is small.

The following 5 numerical schemes will be studied in the following exercises. They all start with $\mathbf{u}_1 = \boldsymbol{\eta}$, then the successive approximations \mathbf{u}_i are defined using (12.4), where the function $\phi(t, \mathbf{u}, h)$ is defined by

- **Euler's method:** $\phi_1(t, \mathbf{u}, h) := f(t, \mathbf{u})$,
- **The backward (or implicit) Euler method:** $\phi_2(t, \mathbf{u}, h) = f(t + h, \mathbf{v})$ where \mathbf{v} is solution of the equation $\mathbf{v} = \mathbf{u} + h f(t + h, \mathbf{v})$.
- **Heun's method:** $\phi_3(t, \mathbf{u}, h) := \frac{1}{2}(f(t, \mathbf{u}) + f(t + h, \mathbf{u} + h f(t, \mathbf{u})))$.
- **The Trapezoidal method:** $\phi_4(t, \mathbf{u}, h) = \frac{1}{2}(f(t, \mathbf{u}) + f(t + h, \mathbf{v}))$ where \mathbf{v} is solution of the equation $\mathbf{v} = \mathbf{u} + \frac{1}{2}h(f(t, \mathbf{u}) + f(t + h, \mathbf{v}))$.
- **Fourth order Runge-Kutta:**

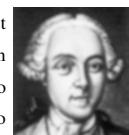
$$\phi_5(t, \mathbf{u}, h) := \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4),$$

where

$$\begin{aligned}\mathbf{k}_1 &:= f(t, \mathbf{u}), & \mathbf{k}_2 &:= f\left(t + \frac{h}{2}, \mathbf{u} + \frac{h}{2}\mathbf{k}_1\right), \\ \mathbf{k}_3 &:= f\left(t + \frac{h}{2}, \mathbf{u} + \frac{h}{2}\mathbf{k}_2\right), & \mathbf{k}_4 &:= f(t + h, \mathbf{u} + h\mathbf{k}_3).\end{aligned}$$

Comment: The backward Euler and the Trapezoidal method are **implicit methods**, a system of equations has to be solved at each step.

Leonhard Euler (1707–1783) was a pioneering Swiss mathematician and physicist who spent most of his working life in St. Petersburg, Russia, and in Berlin, Prussia. He made important discoveries in infinitesimal calculus and graph theory, introducing the notion of a mathematical function. He is also renowned for his work in mechanics, fluid dynamics, optics, and astronomy. A statement attributed to Pierre-Simon Laplace expresses Euler's influence on mathematics: "Read Euler, read Euler, he is the master of us all".



Karl Heun (1859–1929) was a German mathematician who introduced Heun's equation, Heun's functions, and Heun's method. In 1881, after his doctorate he worked as a teacher at an agricultural college in Wehlau. In 1883 he emigrated to England where he taught until 1885. From 1886, he taught at the University of Munich, then in Berlin and Karlsruhe.

Carl David Tolmé Runge (1856–1927) was a German mathematician, physicist and astrophysics. He was co-developer of the Runge Kutta method used to solve ordinary differential equations numerically.



Martin Wilhelm Kutta (1867–1944) was a German mathematician born in Upper Silesia (today Byczyna, Poland). He was professor in Aachen then in Stuttgart. In 1901, he co-developed the Runge-Kutta's method in numerical analysis for differential equations. He is also remembered for his contributions in aerodynamics.



To program these methods, we will follow similar constructions:

Program: If $f \in C([t_1, t_1 + T] \times \mathbb{R}^m, \mathbb{R}^m)$ and $\eta \in \mathbb{R}^m$, the solution of the Cauchy problem (12.1)–(12.2) is approximated by a numerical method using a uniform step length:

$$[\mathbf{t}, \mathbf{u}] = \text{method}(\text{namefunc}, I, \eta, n) \quad (12.5)$$

with inputs the file `namefunc.m` computing the function f , the interval $I = [t_1, t_1 + T]$, the initial condition η and the number of intervals in the partition n . The outputs are the partition \mathbf{t} with $t_j = t_1 + (j - 1)h$ for $j = 1, \dots, n + 1$, where $h = T/n$ and the approximation $\mathbf{u} \in \mathbb{R}^{m \times (n+1)}$ of the solution at the grid points.

Let us notice that with the usual MATLAB tools for example

`[t, u] = ode23(namefunc, I, eta)` the result \mathbf{u} is in $\mathbb{R}^{(n+1) \times m}$.

Stability and convergence: To analyze a method we also consider a modified sequence given by $\mathbf{v}_1 \in \mathbb{R}^m$, $\mathbf{v}_{i+1} := \mathbf{v}_i + h_i \phi(t_i, \mathbf{v}_i, h_i) + \epsilon_i$ where $\epsilon_i \in \mathbb{R}^m$. The scheme (12.4) is **stable** if there exists a constant C independent of the sequences (\mathbf{u}_i) , (\mathbf{v}_i) such that:

$$\max_{1 \leq i \leq n+1} \|\mathbf{u}_i - \mathbf{v}_i\| \leq C \left(\|\mathbf{u}_1 - \mathbf{v}_1\| + \sum_{j=1}^n \|\epsilon_j\| \right).$$

A sufficient condition for the scheme to be stable is that ϕ satisfies a Lipschitz condition in the second variable, i.e., there exists a constant $\Lambda > 0$ and a $H^* > 0$ such that:

$$\forall t \in I, \forall \mathbf{u}, \mathbf{v} \in \mathbb{R}^m, \forall h \in [0, H^*], \|\phi(t, \mathbf{u}, h) - \phi(t, \mathbf{v}, h)\| \leq \Lambda \|\mathbf{u} - \mathbf{v}\|. \quad (12.6)$$

If \mathbf{y} is a solution of (12.1), we set

$$\delta_i^h(\mathbf{y}) := \|\mathbf{y}(t_{i+1}) - \mathbf{y}(t_i) - h_i \phi(t_i, \mathbf{y}(t_i), h_i)\| \text{ for } i = 1, \dots, n.$$

The **consistency error** is defined by $\varepsilon(\mathbf{y}) := \sum_{i=1}^n \delta_i^h(\mathbf{y})$. The scheme is **consistent** if $\lim_{H \rightarrow 0} \max_i \varepsilon(\mathbf{y}) = 0$ for any solution $\mathbf{y} \in C^1(I, \mathbb{R}^m)$. An equivalent condition is $\lim_{H \rightarrow 0} \max_i \left[\frac{\delta_i^h(\mathbf{y})}{h_i} \right] = 0$.

The scheme is **convergent** if $\max_i \|\mathbf{y}(t_i) - \mathbf{u}_i\|$ tends to zero for any solution $\mathbf{y} \in C^1(I, \mathbb{R}^m)$ when $u_1 \rightarrow \eta$ and $H \rightarrow 0$. If the scheme is stable and consistent, then it is convergent.

The scheme is of **order** $p > 0$ if there exists a real number K independent of \mathbf{y} and ϕ such that $\varepsilon(\mathbf{y}) \leq K H^p$ as soon as \mathbf{y} is a C^{p+1} solution of (12.1). An equivalent condition is $\max_i \delta_i^h(\mathbf{y}) = O(H^{p+1})$ as $H \rightarrow 0$.

If \mathbf{y} is a solution of (12.1)–(12.2) with $f \in C^p(I \times \mathbb{R}^m, \mathbb{R}^m)$, if the numerical scheme is stable and of order p , then there exist real numbers Λ and K independent of the partition such that the error satisfies

$$\|\mathbf{y}(t_i) - \mathbf{u}_i\| \leq e^{AT} (\|\boldsymbol{\eta} - \mathbf{u}_1\| + KH^p), \quad i = 1, \dots, n+1.$$

•

12.1 Linear Differential System and Euler's Methods

Exercise 12.1. Sensitivity to the initial condition

For $m = 1$ let us consider the problem

$$\begin{cases} y'(t) &= -150y(t) + 49 - 150t, \quad t \in [0, 1] \\ y(0) &= 1/3 + \varepsilon, \end{cases} \quad (12.7)$$

where $\varepsilon \in \mathbb{R}$ is an error in the initial data.

1. Find the solution $y_{[\varepsilon]}$ of the problem. ▷ *Math Hint*¹ ◁
2. Show that $\|y_{[0]} - y_{[\varepsilon]}\|_\infty \leq |\varepsilon|$.

©Comment: The error in the solution is less than the initial error. ☀

3. Let $h > 0$. If $t, t+h \in [0, 1]$, show that

$$y_{[0]}(t+h) = y_{[0]}(t) + h(-150y_{[0]}(t) + 49 - 150t). \quad (12.8)$$

4. Let $n \in \mathbb{N}$ with $n \neq 0$, $h = \frac{1}{n}$ and $t_i = (i-1)h$, $i = 1, \dots, n+1$. Compute the discrete solution $u_{[\varepsilon]_i}$ for $i = 1, \dots, n+1$ using **Euler's method**.
5. Show that for $i = 1, \dots, n$, $u_{[\varepsilon]_{i+1}} - y_{[0]}(t_{i+1}) = (1 - 150h)(u_{[\varepsilon]_i} - y_{[0]}(t_i))$ and $u_{[\varepsilon]_i} - y_{[0]}(t_i) = (1 - 150h)^{i-1}\varepsilon$ for $i = 1, \dots, n+1$. ▷ *Math Hint*² ◁

©Comment: We notice that $y_{[0]}(t_i) = u_{[0]_i}$. ☀

6. If $n = 50$ and $\varepsilon = 0.01$, compute the error $u_{[\varepsilon]_{n+1}} - y_{[0]}(1)$ at $t = 1$.
7. Give a condition on n to obtain $\max_{i=1, \dots, n+1} |u_{[\varepsilon]_i} - y_{[0]}(t_i)| \leq \varepsilon$.
8. Show how $v_{[\varepsilon]_{i+1}}$ for $1 \leq i \leq n$ can be computed from $v_{[\varepsilon]_i}$ using the **backward Euler method**.
9. Show that for $i = 1, \dots, n$, $v_{[\varepsilon]_i} - y_{[0]}(t_i) = \frac{1}{(1 + 150h)^{i-1}}\varepsilon$. ▷ *Math Hint*³ ◁
10. Give a condition on n to obtain $\max_{1 \leq i \leq n+1} |v_{[\varepsilon]_i} - y_{[0]}(t_i)| \leq \varepsilon$.

¹ The solution is the sum of the general solution of the homogenous equation $y'(t) = -150y(t)$ and a particular solution that is a linear polynomial.

² It is a geometric sequence.

³ Show an equality like (12.8).

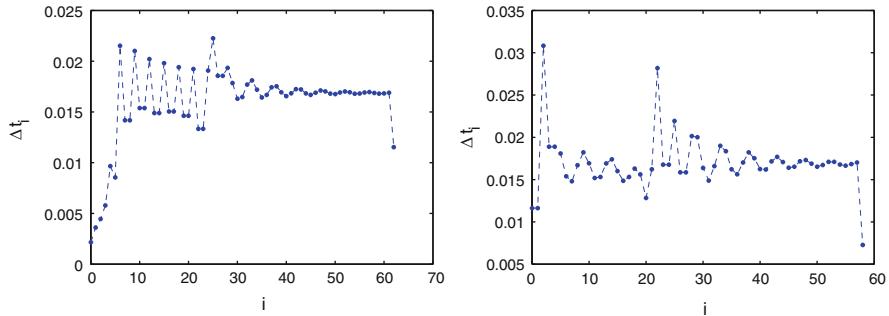


Fig. 12.1 Computation of Δt_i with `ode23` method. *Left:* $\varepsilon = 0.1$, *right:* $\varepsilon = 0.001$.

11. MATLAB: Test the MATLAB tool `ode23` with the initial data $u_1 = 1/3 + 0.01$. Test $[\tau, u] = \text{ode23}('f1', [0, 1], 1/3 + \text{epsilon})$ with $f1(t, u) = -150u + 49 - 150t$, $\varepsilon = 0.1$ and then $\varepsilon = 0.001$. Plot the vector Δt with $\Delta t_i = t_{i+1} - t_i$ in the two cases.

⌚Comment: In Fig. 12.1, the partition is far from uniform and strongly depends on the initial condition. ⌚

NB: We will test Heun's method on problem (12.7) in Exercise 12.5.

Exercise 12.2. A system of linear differential equations

Consider for $t > 0$ the following first-order system

$$\begin{cases} y'_1(t) = 3y_1(t) + y_2(t) - 2t - 4, \\ y'_2(t) = -4y_1(t) - y_2(t) + 3t + 6, \\ y'_3(t) = 4y_1(t) - 8y_2(t) + 2y_3(t) - 16t - 16, \end{cases} \quad \text{with} \quad \begin{cases} y_1(0) = 0, \\ y_2(0) = 3, \\ y_3(0) = 1. \end{cases}$$

1. Show that the problem can be written

$$\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t) + \mathbf{b}(t) \quad (12.9)$$

$$\mathbf{y}(0) = [0, 3, 1]^T, \quad (12.10)$$

where $\mathbf{A} \in \mathbb{R}^{3 \times 3}$, $\mathbf{y}(t) = [y_1(t), y_2(t), y_3(t)]^T$ and $\mathbf{b} \in C(\mathbb{R}, \mathbb{R}^3)$.

2. Find the eigenvalues of \mathbf{A} and show that there exists $\mathbf{P} \in \mathbb{R}^{3 \times 3}$ with first column $[1, 0, 0]^T$ such that $\mathbf{P}^{-1}\mathbf{A}\mathbf{P} = \mathbf{T}$, where $\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 44 & 0 & 2 \end{bmatrix}$ is lower triangular.

▷Math Hint⁴◀

3. Solve the homogeneous system $\mathbf{z}'(t) = \mathbf{T}\mathbf{z}(t)$ and deduce the solutions of $\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t)$. ▷Math Hint⁵◀
4. Find a solution of (12.9) in the form $\mathbf{y}(t) = \alpha t + \beta$ where $\alpha, \beta \in \mathbb{R}^3$.
5. Give the general solution of (12.9)–(12.10).

⁴ See Chap. 3.

⁵ Construct the solution row by row.

Exercise 12.3. The Euler methods for a linear system

We want to approximate the solution of the previous problem,

$$\begin{cases} \mathbf{y}'(t) &= f2(t, \mathbf{y}(t)), t \in [0, 1], \\ \mathbf{y}(0) &= [0, 3, 1]^T. \end{cases}$$

where $f2(t, \mathbf{u}) = \mathbf{A}\mathbf{u} + \mathbf{b}(t)$ with $\mathbf{A} \begin{bmatrix} 3 & 1 & 0 \\ -4 & -1 & 0 \\ 4 & -8 & 2 \end{bmatrix}$ and $\mathbf{b}(t) = \begin{bmatrix} -2t - 4 \\ 3t + 6 \\ -16t - 16 \end{bmatrix}$.

The solution, $\mathbf{y}(t) = \begin{bmatrix} -2e^t + t + 2 \\ -4e^t - t - 1 \\ 40(e^t - e^{2t}) + 2t + 1 \end{bmatrix}$ has been computed in the previous exercise.

1. Write a function file `f2.m` computing the function $f2(t, \mathbf{u}) = \mathbf{A}\mathbf{u} + \mathbf{b}(t)$. Test `f2(0.3, [1 2 3]')`.

```
>> f2(0.3,[1 2 3]')  
ans =  
    0.4000  
    0.9000  
   -26.8000
```

2. Program Euler's method on a uniform partition (see (12.5))

`[t, u]=eulermeth(namefunc, I, eta, n)`. Test:

```
>> [t,u]=eulermeth('f2',[0,1],[0,3,1]',3)  
t =  
    0      0.3333      0.6667      1.0000  
u =  
    0     -0.3333     -0.8889     -1.7407  
  3.0000      4.0000      5.4444      7.4815  
  1.0000    -11.6667    -37.6667    -87.3704
```

3. Find the function $\phi_3(t, \mathbf{u}, h)$ defined in such a way that $\mathbf{v} = \mathbf{u} + h\phi_3(t, \mathbf{u}, h)$ whenever \mathbf{v} is solution of the equation $\mathbf{v} = \mathbf{u} + h f2(t+h, \mathbf{v})$. Write the corresponding function file `phi3.m` which computes the function. Test:

```
>> phi3(2,[1 2 3]',0.1)  
ans =  
  -3.5679  
   7.0247  
 -78.3086
```

4. Program the backward Euler method on an uniform partition.

`[t, v]=backeulermeth(namefunc, I, eta, n)`. Test:

```
>> [t,v]=backeulermeth('phi3',[0,1],[0,3,1]',3)  
t =  
    0      0.3333      0.6667      1.0000  
v =  
    0     -0.6667     -1.8333     -3.7500  
  3.0000      4.6667      7.3333     11.5000  
  1.0000    -58.3333    -267.6667    -942.0000
```

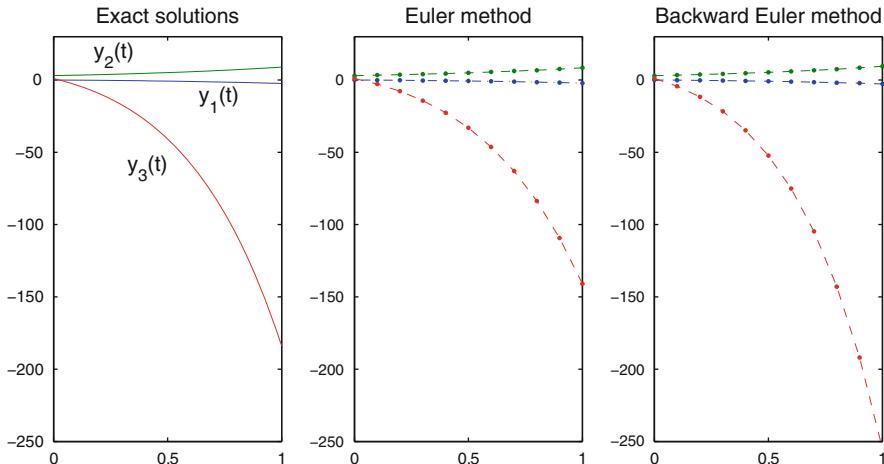


Fig. 12.2 Exact solutions and Approximations for $n = 10$

5. Write a program `lineardiffsystem.m` that for a given n computes the approximations by the two methods: Euler and backward Euler, then plots the three graphs. The results are shown in Fig. 12.2.
6. Write a program to study the errors when modifying n . Start with an array $arrn = 100 : 100 : 200$ to compute the corresponding arrays $arrerr1$ for Euler's method and $arrerr2$ for backward Euler method, then plot the two graphs $\log(arrerri)$ depending on $\log(arrn)$; the first error will be computed with $error1 = \sum_{\ell=1}^3 \|y_\ell - u(\ell, :) \|_\infty$ and a similar expression for the second error.

12.2 Other Examples of One Step Methods

Exercise 12.4. Stability and order

We consider the Cauchy problem (12.1)–(12.2) where $\eta \in \mathbb{R}^m$ and $f \in C(I \times \mathbb{R}^m, \mathbb{R}^m)$ satisfies the Lipschitz condition in its second variable (see (12.3)). Thus the differential problem has a unique solution.

For the numerical approximation we consider the partition

$t_1 < t_2 < \dots < t_{n+1} = t_1 + T$, set $h_i = t_{i+1} - t_i$, and $H = \max_{1 \leq i \leq n} h_i$.

1. The numerical scheme is Heun's method defined by $\mathbf{u}_1 = \eta$ and

$$\mathbf{u}_{i+1} = \mathbf{u}_i + h_i \phi(t_i, \mathbf{u}_i, h_i), \quad i = 1, \dots, n, \quad (12.11)$$

where $\phi(t, \mathbf{u}, h) = \frac{1}{2}(f(t, \mathbf{u}) + f(t + h, \mathbf{u} + hf(t, \mathbf{u})))$.

- Show that Heun's method is stable. \triangleright Math Hint⁶ \triangleleft
 - We set $\delta^h(\mathbf{y}) = \|\mathbf{y}(t+h) - \mathbf{y}(t) - h\phi(t, \mathbf{y}(t), h)\|$, where $\mathbf{y}(t) \in C^3(\mathbb{R}, \mathbb{R}^m)$ is a solution of (12.1). Show that $\delta^h(\mathbf{y}) = O(h^3)$. \triangleright Math Hint⁷ \triangleleft
 - Deduce the convergence and the order of the scheme.
2. The numerical scheme is now defined by $\mathbf{u}_1 = \eta$ and

$$\mathbf{u}_{i+1} = \mathbf{u}_i + h_i \psi(t_i, \mathbf{u}_i, h_i), \quad i = 1, \dots, n, \quad (12.12)$$

where $\mathbf{v} = \psi(t, \mathbf{u}, h)$ is such that $\mathbf{v} = \mathbf{u} + h f \left(t + \frac{h}{2}, \frac{1}{3}\mathbf{u} + \frac{2}{3}\mathbf{v} \right)$. Show that this implicit numerical scheme is stable. \triangleright Math Hint⁸ \triangleleft

Exercise 12.5. Heun's method for a linear equation

We return to the problem (12.7) studied in Exercise 12.1, with unique solution $y(t) = -t + 1/3 + \varepsilon e^{-150t}$.

- Write a function file `f1.m` that computes the function

$$f1(t, u) = -150u + 49 - 150t.$$

- Program Heun's method on an uniform partition (see (12.5) for inputs and outputs)

`[t, u] = Heun(namefunc, I, etat, n)`. Test:

```
>> [t, u] = Heun('f1', [0, 1], 1/3 + 0.01, 4)
t =
      0      0.2500      0.5000      0.7500      1.0000
u =
  1.0e+009 *
      0.0000      0.0000      0.0030      1.9748
```

- Write a program that plots the graphs of the approximation and of the exact solution and computes the error: $\max_{i=0, \dots, n} |u_i - y(t_i)|$ where $y(t)$ is the exact solution. Save as `Heungraph.m`. Test with $n = 40, 73, 75, \dots$ and $\varepsilon = 0.01$.

```
>> Heungraph
n =
  40
error =
  1.8316e+023
```

Comment: On the graph in Fig. 12.3, we see that, again, as in Euler's method, the error can be huge. \bullet

- Write a program `Heunerror.m` to study the error for different values of n . Beginning with an array `arrn` the program will compute the corresponding

⁶ Use (12.6) and the Lipschitz condition of f .

⁷ Use $f(t, \mathbf{y}(t)) = \mathbf{y}'(t)$ and introduce $\mathbf{y}'(t+h) - f(t+h, \mathbf{y}(t+h))$ in $\phi(t, \mathbf{y}(t), h)$.

⁸ Use (12.6) and the Lipschitz condition of f .

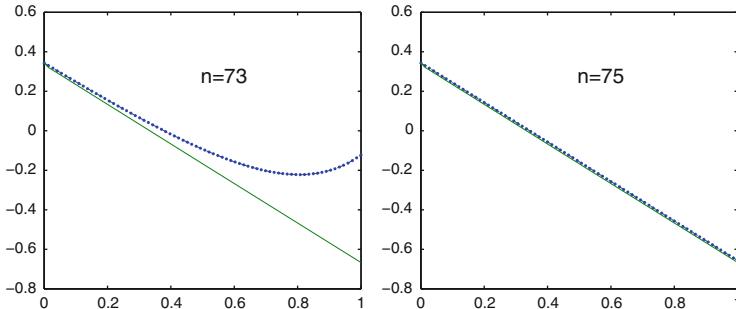


Fig. 12.3 Exact solution and approximation by Heun's method

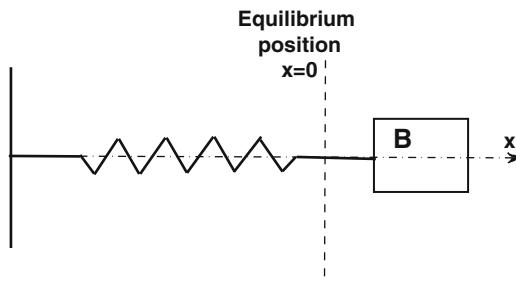


Fig. 12.4 Wall-Spring-Body

arrerror and plot $\ln(\text{arrerror})$ depending on $\ln(\text{arrn})$. What seems to be the order of the scheme. Test $\text{arrn} = 5000 : 10 : 5100$ and $\varepsilon = 0.01$.

Exercise 12.6. Movements of a body attached to a spring

A body B of mass m resting on a horizontal, frictionless plane is attached to an ordinary spring while the other end of the spring is attached to a fixed wall (see Fig. 12.4). We assume that all movements of B occur along an x -axis normal to the wall. The spring will exert a force on B if it is stretched ($x > 0$) or compressed ($x < 0$). The force tends to bring B back to its equilibrium position $x = 0$. We assume that the force is given in the form $-kx + x^3$, where $k > 0$ is a spring constant. If $x(t)$ is the position of B at time t , by Newton's second law, we obtain

$$mx''(t) = -kx(t) + x^3(t).$$

In the following, we assume that $m = 1$, and we write $z(t) = x'(t)$ for the velocity so that we have the system

$$\begin{cases} x'(t) &= z(t), \quad t \in [0, b] \\ z'(t) &= -kx(t) + [x(t)]^3 \end{cases} \quad (12.13)$$

with initial conditions $x(0) = \alpha$ and $z(0) = x'(0) = \beta$.

- If we define the kinetic energy of B as $T(t) := \frac{1}{2}z(t)^2$ and the potential energy of the spring as $V(t) := \frac{1}{2}kx(t)^2 - \frac{1}{4}x(t)^4$, show that $E(t) := T(t) + V(t) = E$ where E is a constant independent of t . *Math Hint*⁹
- Write (12.13) in the form $\mathbf{y}'(t) = f2(t, \mathbf{y}(t))$ where $f : [0, b] \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$. In the following we will take $k = 1$.
- Programs:** Write the corresponding function file `f2.m`, then compute an approximation of \mathbf{y} using Heun's method on an uniform partition. Test:

```
>> f2(1,[1,3])
ans =
    3
    0
```

- Let $Eapprox(i)$ be the corresponding approximation of $E(t_i) = T(t_i) + V(t_i)$. Write a program `springerror.m` to study the error for different values of n where

$$\text{error} = \max_{i=2, \dots, n+1} |Eapprox(i) - Eapprox(1)|.$$

Beginning with an array `arrn` the program will compute the corresponding `arrerror`. Test on $I = [0, 19]$ and $\eta = [0.1; 1.5]$ with the array $arrn = 100 : 10 : 200$.

Exercise 12.7. 4th order Runge-Kutta for an ODE

We want to solve numerically the problem

$$\begin{cases} y^{(4)}(t) + y''(t)y^2(t) = \frac{1}{2}\sin(2t)\cos t, & t \in I = [0, 2\pi] \\ y(0) = 0, y'(0) = 1, y''(0) = 0, y^{(3)}(0) = -1. \end{cases} \quad (12.14)$$

and compare it with the exact solution $y(t) = \sin t$.

- Let $\mathbf{y}(t) = [y(t), y'(t), y''(t), y^{(3)}(t)]^T$. Define a function $f3 : I \times \mathbb{R}^4 \rightarrow \mathbb{R}^4$ such that (12.14) can be written

$$\begin{cases} \mathbf{y}'(t) = f3(t, \mathbf{y}(t)), & t \in I = [0, 2\pi] \\ \mathbf{y}(0) = \eta. \end{cases} \quad (12.15)$$

- Program** the corresponding function file `f3.m`. Test

```
>> z=f3(1,[1,2,-1,3])
z =
    2.0000
   -1.0000
    3.0000
   1.2456
```

⁹ Compute $E'(t)$

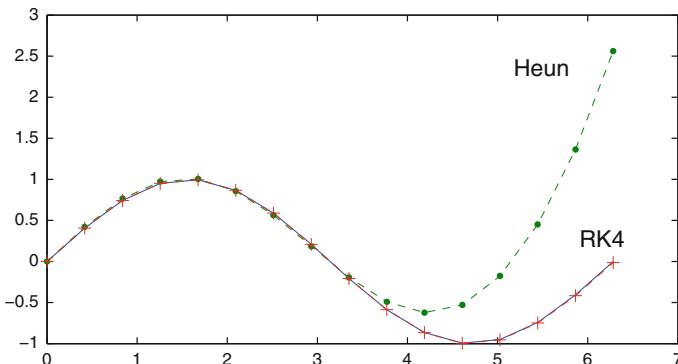


Fig. 12.5 Exact solution and Approximations by Heun's and 4th order Runge Kutta methods

3. Program the 4th order Runge-Kutta's method on a uniform partition, $[t, u] = \text{RK4}(\text{namefunc}, I, \text{eta}, n)$. See (12.5) for the input and output parameters.

```
>> [t, u]=RK4('f3',[0,2*pi],[0,1,0,-1]',4)
t =
      0    1.5708    3.1416    4.7124    6.2832
u =
      0    0.9248    0.0404   -0.5987    1.0722
    1.0000    0.0370   -0.8027    0.1924    1.8681
      0   -0.8815   -0.0095    1.0200    0.9540
   -1.0000   -0.0715    0.9806    0.3143   -0.6235
```

4. Write a program plotting the approximations with Runge Kutta's and Heun's methods and also the exact solution of (12.14). Save as `order4graph.m`. Test with $n = 15$. See Fig. 12.5.
 5. Write a program `RK4error.m` to study the error for different values of n . Beginning with an array `arrn` the program will compute the corresponding `arrerror` and plot $\ln(\text{arrerror})$ depending on $\ln(\text{arrn})$. Test $\text{arrn} = 100 : 110$. What seems to be the order of the scheme?

Exercise 12.8. Extending the solution of an ODE

We want to solve the problem

$$\begin{cases} (1-t^2)y'(t) + (2t-1)y(t) = 1, & t \in I \\ y(0) = 2/3, \end{cases} \quad (12.16)$$

where I is an interval containing 0.

1. Find the solutions of the homogeneous equation $(1-t^2)y'(t) + (2t-1)y(t) = 0$ on an interval J such that $J \cap \{-1, 1\} = \emptyset$. ▷ *Math Hint*¹⁰ ◁

¹⁰ Integrate $y'(t)/y(t)$.

2. Find a particular polynomial solution of $(1 - t^2)y'(t) + (2t - 1)y(t) = 1$ and give the general solution of $(1 - t^2)y'(t) + (2t - 1)y(t) = 1$ on an interval J such that $J \cap \{-1, 1\} = \emptyset$. ▷ *Math Hint*¹¹ ◁
3. Show that the general solution can be extended to all of \mathbb{R} and deduce the solution of (12.16) on I .
4. **Programs:** Program a function file *f4.m* such that

$$(1 - t^2)y'(t) + (2t - 1)y(t) = 1 \Leftrightarrow y'(t) = f4(t, y(t)).$$

5. Program the 4th order Runge-Kutta's method on an uniform partition (or use the program in Exercise 12.7). Test

```
>> [t , u]=RK4( 'f4' ,[ 0 , 0.5 ] ,2/3 ,4)
t =
0      0.1250      0.2500      0.3750      0.5000
u =
0.6667    0.8766    1.0853    1.2851    1.4657
```

6. Write a program plotting the discrete and exact solutions of (12.16). Save as *extendgraph.m*. Test with $I = [0, 0.90]$ then $I = [0, \pi/2]$ for $n = 20$.

12.3 Predictor-Corrector

A predictor–corrector method proceeds in two steps. First, the prediction step calculates an approximation of the desired quantity, and then the corrector step refines the initial approximation using another method.

Exercise 12.9. Test of Predictor-Corrector methods

For the Cauchy problem (12.1)–(12.2), the numerical approximation is defined on the partition $t_1 < t_2 < \dots < t_{n+1} = t_1 + T$, and we set $h_i = t_{i+1} - t_i$.

We will test two methods:

Euler's method with the trapezoidal rule: $u_1 = \eta$, and for $i = 1, \dots, n$

$$\begin{cases} v &= u_i + h_i f(t_i, u_i), \\ u_{i+1} &= u_i + \frac{h_i}{2} (f(t_i, u_i) + f(t_{i+1}, v)), \end{cases} \quad (12.17)$$

Euler's method using the trapezoidal rule twice: $u_1 = \eta$, and for $i = 1, \dots, n$

$$\begin{cases} v &= u_i + h_i f(t_i, u_i), \\ w &= u_i + \frac{h_i}{2} (f(t_i, u_i) + f(t_{i+1}, v)), \\ u_{i+1} &= u_i + \frac{h_i}{2} (f(t_i, u_i) + f(t_{i+1}, w)), \end{cases} \quad (12.18)$$

¹¹ $p \in \mathbb{P}_2$.

for the problem (12.14) of Exercise 12.7 that is transformed into a system of 4 differential equations of first order. The function such that $\mathbf{y}'(t) = \mathbf{f}3(t, \mathbf{y}(t))$ on $[0, 2\pi]$ is defined by $\mathbf{f}3(t, \mathbf{u}) = [u_2, u_3, u_4, 1/2 \sin(2t) \cos t - u_3 u_1^2]^T$ and initial conditions are transformed into $\boldsymbol{\eta} = [0, 1, 0, -1]^T$. The first component of the exact solution is $y_1(t) = \sin(t)$.

1. Write a function file `f3.m` that computes the corresponding function. (See Exercise 12.7).
2. Program the function files `predictcorrect1.m` and `predictcorrect2.m` corresponding to (12.17) and (12.18). See (12.5) for the input and output parameters. Test:

```
>> [t, u] = predictcorrect1('f3', [0, 2*pi], [0, 1, 0, -1]', 4)
t =
    0      1.5708      3.1416      4.7124      6.2832
u =
1.0e+003 *
    0      0.0016     -0.0007      0.0069      0.0376
    0.0010     -0.0002     -0.0002      0.0139      0.0143
    0     -0.0016      0.0064      0.0072     -0.4310
   -0.0010      0.0020      0.0032     -0.0088      4.0068
```

```
>> [t, u] = predictcorrect2('f3', [0, 2*pi], [0, 1, 0, -1]', 4)
t =
    0      1.5708      3.1416      4.7124      6.2832
u =
1.0e+007 *
    0      0.0000      0.0000      0.0000      0.0006
    0.0000     -0.0000      0.0000     -0.0000      0.0008
    0      0.0000      0.0000     -0.0000     -0.9429
   -0.0000      0.0000     -0.0000      0.0006     -1.6022
```

3. Write a program that plots the graphs of the two approximations and also the exact solution of (12.14). Save as `pcgraph.m`. Test with $n = 15$. Graphs in Fig. 12.6.
4. Write a program `pcerror.m` to study the errors by the two methods for different values of n . Test starting with $arrn = 1000 : 10 : 1100$ and give a conclusion of the order.

12.4 Application of ODE Solvers to the Shooting Method

The shooting method is used to solve a boundary value problem by reducing it to the solution of an initial value problem. Consider the problem of finding a solution $y \in C^2([0, 1])$ such that

$$(\mathcal{SP}) \quad \begin{cases} y''(t) &= f(t, y(t), y'(t)), t \in [0, 1] \\ y(0) &= 1, y(1) = a \end{cases} \quad (12.19)$$

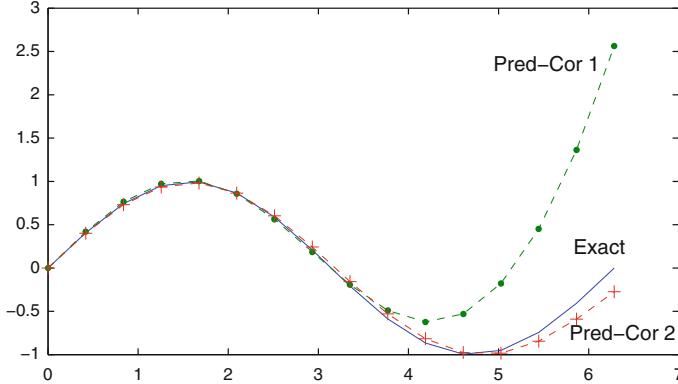


Fig. 12.6 Exact solution and the approximations by the two predictor–corrector methods

If for any $s \in \mathbb{R}$, the Cauchy problem

$$(\mathcal{CP}_s) \begin{cases} y''(t) &= f(t, y(t), y'(t)) \\ y(0) &= 1, \quad y'(0) = s. \end{cases} \quad (12.20)$$

has a unique solution $y_{[s]}$, the problem (12.19) is solved as soon as $s \in \mathbb{R}$ satisfies $y_{[s]}(1) = a$. The shooting method gives an algorithm to generate a sequence (s_p) that hopefully converges to an s such that $y_{[s]}(1) = a$.

Exercise 12.10. Shooting method for a linear problem

We write $y, y_{[s]}$ for the respective solutions of the problems

$$(\mathcal{SP}) \begin{cases} y''(t) = 1 - 2t \cos t - y(t) + ty'(t), \\ y(0) = 1, \quad y(1) = a. \end{cases}$$

$$(\mathcal{CP}_s) \begin{cases} y''(t) = 1 - 2t \cos t - y(t) + ty'(t), \\ y(0) = 1, \quad y'(0) = s. \end{cases}$$

1. Show that there exists $\lambda \in \mathbb{R}$ such that $y = \lambda y_{[0]} + (1 - \lambda) y_{[1]}$.
2. Transform (\mathcal{CP}_s) into a system of first order

$$(\mathcal{DS}_s) \begin{cases} \mathbf{y}'(t) &= f(t, \mathbf{y}(t)), \\ \mathbf{y}(0) &= [1, s]^T. \end{cases}$$

where $f(t, \mathbf{u}) \in C([0, 1] \times \mathbb{R}^2)$.

3. **Numerical approximation:** On a partition $0 = t_1 < t_2 < \dots, t_{n+1} = 1$ of $[0, 1]$, we choose a numerical scheme to approximate the solution of (\mathcal{DS}_s) at the points t_i and we write $\mathbf{u}_{[s]} \in \mathbb{R}^{2 \times (n+1)}$ for the approximation, i.e. $y_{[s]}(t_i) \approx u_{[s]}(1, i)$. In the program, the array $\mathbf{u}_{[0]}$ (resp. $\mathbf{u}_{[1]}$) will be written $U0$ (resp. $U1$). Finally, the solution y of (\mathcal{SP}) at t_i is approximated by

$$u_i = \ell \times U0(1, i) + (1 - \ell) \times U1(1, i), \quad i = 1, \dots, n + 1,$$

where ℓ is chosen such that

$$\ell \times U0(1, n + 1) + (1 - \ell) \times U1(1, n + 1) = a.$$

- a. For the approximation of (\mathcal{DS}_s) , write the corresponding function file `f.m`.

```
>> f(1, [-1 2])
ans =
    2.0000
    2.9194
```

- b. The solution of (\mathcal{DS}_s) is approximated by Euler's method on an uniform partition. Write a MATLAB function file

`[t, U] = eulermeth(namefunc, interv, eta, n)` (see (12.5) for the parameters and Exercise 12.3 for the solution). Test

```
>> [t, u] = eulermeth('f', [0, 1], [1; 0], 4)
t =
    0    0.2500    0.5000    0.7500    1.0000
u =
    1.0000    1.0000    1.0000    0.9697    0.8808
        0         0    -0.1211   -0.3556   -0.6891
```

- c. Program the shooting method `[t, u] = shooteuler(namefunc, n, a)` with inputs the file `namefunc` computing $f(t, u)$, the number of subintervals n , and the boundary condition on the right a . The outputs are the arrays `t` and `u` of length $n + 1$. Test

```
>> [t, u] = shooteuler('f', 4, 2 * (1 + sin(1)))
t =
    0    0.2500    0.5000    0.7500    1.0000
u =
    1.0000    1.7005    2.4011    3.0713    3.6829
```

- d. The exact solution of (\mathcal{SP}) is $y(t) = t + 2 \sin t + 1$ for $a = 2(1 + \sin 1)$. For a given n , plot the discrete and exact solutions. Also compute the error $\max_j \|y(t_j) - u_j\|$. Save as `errorshooteuler1.m`. Test with $n = 4$. The graph is in Fig. 12.7.

```
>> errorshooteuler1
error =
    0.0578
```

- e. Write a program to study the error. Starting with an array `arrn = [5, 10, 20, 50, 100, 200]`, compute the corresponding array `arrerror`, then plot $\log(arrerror)$ depending on $\log(arrn)$ and estimate the order of the method. Save as `errorshooteuler2.m`.
- f. Replace Euler's method by the 4th order Runge-Kutta method. What seems to be the order now?

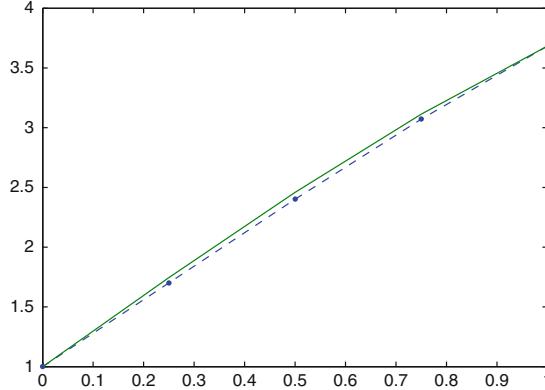


Fig. 12.7 Exact and discrete solutions with Euler's method

Exercise 12.11. Another Runge-Kutta method, RK_3 .

For a function $f \in C([a, b] \times \mathbb{R}^m)$ that satisfies a Lipschitz condition in its second variable, let the function $\mathbf{y}(t) = [y_1(t), y_2(t), \dots, y_m(t)]^T$ from $[a, b]$ to \mathbb{R}^m be the solution of the Cauchy problem

$$\begin{cases} \mathbf{y}'(t) = f(t, \mathbf{y}(t)), & t \in [a, b] \\ \mathbf{y}(a) = \boldsymbol{\eta} \in \mathbb{R}^m. \end{cases} \quad (12.21)$$

For $n \in \mathbb{N}$, we set $h = \frac{b-a}{n}$ and we define the uniform partition $\mathbf{t} \in \mathbb{R}^{n+1}$ of $[a, b]$, with $t_i = [a + (i-1)h]$ for $i = 1, \dots, n+1$. To approximate the solution of (12.21), $\mathbf{u}_i \approx \mathbf{y}(t_i) \in \mathbb{R}^m$, we define the following 3rd order Runge-Kutta method

$$\begin{cases} \mathbf{u}_1 = \boldsymbol{\eta} \\ \mathbf{u}_{i+1} = \mathbf{u}_i + \frac{h}{6}(\mathbf{k}_1 + 4\mathbf{k}_2 + \mathbf{k}_3), & i = 1, \dots, n, \\ \text{with } \begin{cases} \mathbf{k}_1 = f(t_i, \mathbf{u}_i), \\ \mathbf{k}_2 = f(t_i + \frac{h}{2}, \mathbf{u}_i + \frac{h}{2}\mathbf{k}_1), \\ \mathbf{k}_3 = f(t_i + h, \mathbf{u}_i - h\mathbf{k}_1 + 2h\mathbf{k}_2). \end{cases} \end{cases} \quad (12.22)$$

We will study this method on the example:

$$\begin{cases} y''(t) = -y(t), & t \in [0, \pi] \\ y(0) = 0, \quad y'(0) = 1, \end{cases} \quad (12.23)$$

which is rewritten as a system of order 1. Setting $\mathbf{y}(t) = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$, we obtain

$$\begin{cases} \mathbf{y}'(t) = f(t, \mathbf{y}(t)), & t \in [0, \pi] \\ \mathbf{y}(0) = [0, 1]^T. \end{cases} \quad (12.24)$$

where $f : [0, \pi] \times \mathbb{R}^2 \rightarrow \mathbb{R}^2$.

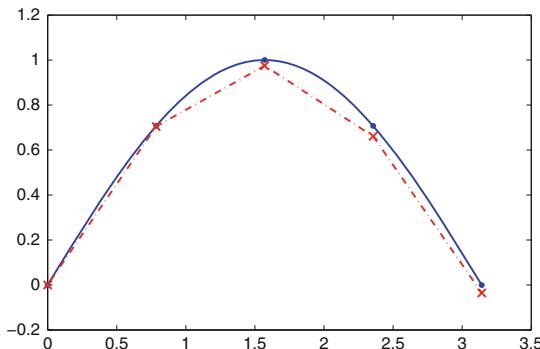


Fig. 12.8 Exact and discrete solution by the RK3 method

1. Write the corresponding function file `f.m`. Test:

```
>> f(1,[1 2])
ans =
    2
   -1
```

2. Write a function file `RK3.m` computing the approximation using (12.22) on a uniform partition (see (12.5) for the parameters). Test

```
>> [t,u]=RK3('f',[0,pi],[0;1],4)
t =
      0     0.7854     1.5708     2.3562     3.1416
u =
      0     0.7047     0.9746     0.6612    -0.0356
    1.0000     0.6916    -0.0183    -0.6994    -0.9496
```

3. Write a file `RK3error1.m` that for a given n computes the approximation, plot exact and discrete solutions, ▷ *Math Hint*¹² ◃ then evaluates

$$err1 = \max_{1 \leq i \leq n+1} |u(1,i) - \sin(t_i)|.$$

Test with $n=5$.

```
>> RK3error1
err =
  0.0459
```

See Fig. 12.8 for the graphs.

4. Study the error and give the computed order of the method.
Start with $arrn = 20 : 10 : 100$. Save as `RK3error2`.

¹² The exact solution is $y(t) = \sin(t)$.

Exercise 12.12. Shooting method for a non-linear problem

We want an approximation of the solution of the problem

$$(\mathcal{SP}) \begin{cases} y''(t) = \frac{3}{2}y^2(t), & t \in [0, 1], \\ y(0) = 4, & y(1) = 1 \end{cases} \quad (12.25)$$

We will find two approximations. For $s \in \mathbb{R}$, we define the Cauchy problem

$$(\mathcal{CP}_s) \begin{cases} y''(t) = \frac{3}{2}y^2(t), & t \in [0, 1], \\ y(0) = 4, & y'(0) = s. \end{cases} \quad (12.26)$$

The solution of (12.26) is written $y_{[s]}$. With the shooting method, we are looking for values $s \in \mathbb{R}$ such that $y_{[s]}(1) = 1$ which gives a solution of (12.25).

In the following, for $s \in \mathbb{R}$ and n a positive integer, we write $\mathbf{u}_{[s]} \in \mathbb{R}^{2 \times (n+1)}$ for the matrix of approximations $\begin{bmatrix} u_{[s]}(1, j) \\ u_{[s]}(2, j) \end{bmatrix} \approx \begin{bmatrix} y_{[s]}(t_j) \\ y'_{[s]}(t_j) \end{bmatrix}$ for $j = 1, \dots, n + 1$ computed by RK3, so that in particular $u_{[s]}(1, n + 1) \approx y_{[s]}(1)$. Now by the shooting method, we look for a solution $s \in \mathbb{R}$ of

$$w(s, n) := u_{[s]}(1, n + 1) - 1 = 0, \quad (12.27)$$

We compute a sequence of approximations to the solution σ of (12.27) by the **secant method**: from a value s_1 not too far from σ and a small $h > 0$, we define the finite sequence (s_k) for $k = 1, \dots, n_0$ by

$$\begin{aligned} s_2 &= s_1 - \frac{w(s_1, n)}{w(s_1 + h, n) - w(s_1, n)} h \\ s_{k+1} &= s_k - \frac{w(s_k, n)}{w(s_k, n) - w(s_{k-1}, n)} (s_k - s_{k-1}), \quad k = 2, \dots, n_0 - 1 \end{aligned}$$

and the approximation of σ is s_{n_0} .

- To approximate the solution of (\mathcal{CP}_s) in (12.26) by the 3rd order Runge-Kutta method the second order differential equation is rewritten into a differential system of order 1: $\mathbf{y}'(t) = f(t, \mathbf{y}(t))$ where $\mathbf{y}(t) = \begin{bmatrix} y(t) \\ y'(t) \end{bmatrix}$. Write the corresponding function file `f.m`. Test

```
>> f(1, [3; 2])
ans =
    2.0000
   13.5000
```

- Write a function file `w.m` with inputs the real number s and the integer n , and outputs $u_{[s]}(1, n + 1) - 1$ where $\mathbf{u}_{[s]} \in \mathbb{R}^{2 \times (n+1)}$ is the discrete solution of (\mathcal{CP}_s) computed by (12.22) (cf. `RK3.m` in Exercise 12.11, Question 2) on an uniform partition on $[0, 1]$. Test `w(0, 20)`.

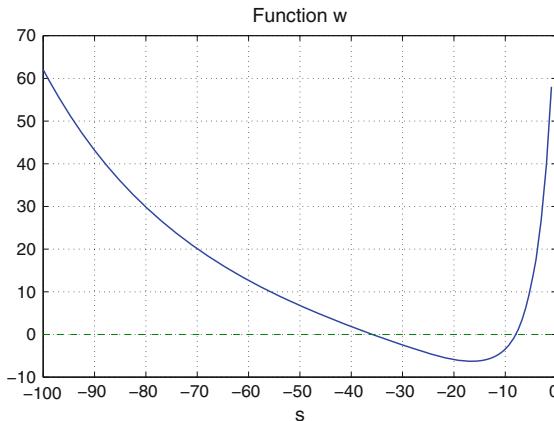


Fig. 12.9 Graph of the function $w(s, 20)$

```
>> w(0,20)
ans =
85.1206
```

3. Plot the curve $w(s, n)$ for s in $[-100, -1]$ and $n = 20$. Starting from an array $s = [-100 : -1]$, the program will compute the corresponding values w and then will plot s, w . Save as `plotw.m`.
On the graph of the curve (Fig. 12.9), we see that w has two roots, both are solutions of (12.27). We will call these roots σ_1 and σ_2 with $\sigma_1 < \sigma_2$.
4. For $i = 1, 2$, give a localization of the roots in intervals $I_i = [10z_i, 10(z_i + 1)]$ with $z_i \in \mathbb{Z}$ for $n = 20$.
5. Compute an approximation of σ_1 using the secant method, starting with $s_1 = -36$, $h = 0.1$, $n_0 = 5$ and $n = 20$. Save as `Approxsigma1.m`.
6. Continue the previous program with the approximation of σ_2 , starting from $s_1 = -7$. Save as `Approxsigmas.m`
7. Continue the previous program with the computation of the two corresponding approximations of the solutions of (12.25) and their graphs. Save as `Approxsol.m`. Test with $n = 20$. ◁ MATLAB Hint¹³ ▷
8. Show that the exact solution of (\mathcal{CP}_s) satisfies $(y'_{[s]})^2 - y_{[s]}^3 - s^2 + 64 = 0$ for any $s \in \mathbb{R}$.
9. Continue the previous program with the computation of

$$\text{err} = \max_{1 \leq i \leq n+1} |u_{[\sigma_1]}(2, i)^2 - u_{[\sigma_1]}(1, i)^3 - \sigma_1^2 + 64|. \quad (12.28)$$

¹³ Use `grid on`.

for a given n . Test with $n = 100$, $n_0 = 5$, $h = 0.1$ and $s_1 = -36$. Save as `error1.m`.
 <MATLAB Hint¹⁴>

```
>> error1
sigma1 =
-35.8587
err =
0.0286
```

10. Study the error when modifying n . Start with the array `arrn= [100 : 110]` and compute the corresponding errors `arrerror`. What is your estimation of the order? Save as `error2.m`. Test with $n_0 = 5$, $h = 0.1$ and $s_1 = -36$.

12.5 Solutions

12.5.1 Linear Differential System and Euler's Methods

Solutions of Exercise 12.1

Let $f1(t, u) = -150u + 49 - 150t$.

- The homogeneous equation $y'(t) = -150y(t)$ has the family of solutions $y(t) = \lambda e^{-150t}$ with $\lambda \in \mathbb{R}$. A particular solution of $y'(t) = f1(t, y(t))$ is $y(t) = -t + 1/3$. Thus the general solution of the linear differential equation is $y(t) = \lambda e^{-150t} - t + 1/3$. The solution of (12.7) is then $y_{[\varepsilon]}(t) = \varepsilon e^{-150t} - t + 1/3$ for $\varepsilon \geq 0$.
- $|y_{[0]}(t) - y_{[\varepsilon]}(t)| = |\varepsilon e^{-150t}| \leq |\varepsilon|$ for $t \geq 0$. Hence $\|y_{[0]} - y_{[\varepsilon]}\|_\infty \leq |\varepsilon|$.
- We have $y_{[0]}(t+h) - y_{[0]}(t) = hy'_{[0]}(t)$ since $y_{[0]}$ is an affine function. Moreover, $y_{[0]}$ is a solution of the differential equation, so that $y'_{[0]}(t) = -150y_{[0]}(t) + 49 - 150t$. From these two formulas, we deduce that

$$y_{[0]}(t+h) = (1 - 150h)y_{[0]}(t) + h(49 - 150t). \quad (12.29)$$

- Euler's method on a uniform partition applied to (12.7) takes the form

$$\begin{aligned} u_{[\varepsilon]0} &= \varepsilon + 1/3, \\ u_{[\varepsilon]i+1} &= u_{[\varepsilon]i} + hf1(t_i, u_{[\varepsilon]i}) = (1 - 150h)u_{[\varepsilon]i} + h(49 - 150t_i). \end{aligned}$$

- By subtracting (12.29) at $t = t_i$ from the previous equality, we obtain

$$u_{[\varepsilon]i+1} - y_{[0]}(t_{i+1}) = (1 - 150h)(u_{[\varepsilon]i} - y_{[0]}(t_i))$$

which gives a geometric sequence with first term $u_{[\varepsilon]1} - y_{[0]}(0) = \varepsilon$. We deduce that $u_{[\varepsilon]i} - y_{[0]}(t_i) = (1 - 150h)^{i-1}\varepsilon$ for $i = 1, \dots, n + 1$.

¹⁴ Since n is modified, σ_1 has to be recomputed.

6. If $n = 50$ and $\varepsilon = 0.01$, then $150h = 3$ and the error satisfies

$$|u_{[\varepsilon]_{n+1}} - y_{[0]}(1)| = |(-2)^{50} \times 0.01| \approx 1.126 \times 10^{13}.$$

Comment: For small values of n , the error can be huge. ☺

7. $\max_{1 \leq i \leq n+1} |u_{[\varepsilon]_i} - y_{[0]}(t_i)| \leq |\varepsilon|$ provided $-1 \leq 1 - 150h \leq 1$.

With $h = 1/n > 0$, the condition is $n \geq 75$.

8. The backward Euler method gives

$$\begin{aligned} v_{[\varepsilon]_0} &= \varepsilon + 1/3, \\ v_{[\varepsilon]_{i+1}} &= v_{[\varepsilon]_i} + h f_1(t_{i+1}, v_{[\varepsilon]_{i+1}}) \\ &= v_{[\varepsilon]_i} + h(49 - 150(t_i + h) - 150v_{[\varepsilon]_{i+1}}), \end{aligned}$$

from which we deduce that

$$(1 + 150h)v_{[\varepsilon]_{i+1}} = v_{[\varepsilon]_i} + h(49 - 150(t_i + h)). \quad (12.30)$$

9. We have $y_{[0]}(t + h) = y_{[0]}(t) + hy'_{[0]}(t + h)$ since $y_{[0]}$ is affine. This gives $y_{[0]}(t + h) = y_{[0]}(t) + h(49 - 150(t + h) - 150y_{[0]}(t + h))$ or equivalently

$$(1 + 150h)y_{[0]}(t) = y_{[0]}(t) + h(49 - 150(t + h)).$$

When subtracting this equation at $t = t_i$ from (12.30) we obtain $(1 + 150h)(v_{[\varepsilon]_{i+1}} - y_{[0]}(t_{i+1})) = v_{[\varepsilon]_i} - y_{[0]}(t_i)$ for $i = 1, \dots, n+1$, and finally

$$v_{[\varepsilon]_i} - y_{[0]}(t_i) = \frac{\varepsilon}{(1 + 150h)^{i-1}}, \quad i = 1, \dots, n+1.$$

10. So that for any n , we have $\max_{1 \leq i \leq n+1} |v_{[\varepsilon]_i} - y_{[0]}(t_i)| \leq |\varepsilon|$.

11. MATLAB :

```
[t, u]=ode23('f1',[0 1],1/3+0.1)
n=length(t); plot(0:n-2,t(2:n)-t(1:n-1),'.--')
xlabel('i')
ylabel('\Delta t_i')
```

Solutions of Exercise 12.2

1. The equations can be written $\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t) + \mathbf{b}(t)$ with

$$\mathbf{A} = \begin{bmatrix} 3 & 1 & 0 \\ -4 & -1 & 0 \\ 4 & -8 & 2 \end{bmatrix}, \quad \mathbf{b}(t) = \begin{bmatrix} -2t - 4 \\ 3t + 6 \\ -16t - 16 \end{bmatrix},$$

and the initial conditions are written $\mathbf{y}(0) = [0, 3, 1]^T$.

2. The characteristic polynomial of \mathbf{A} is $p(\lambda) = (2 - \lambda)(\lambda - 1)^2$.

For the eigenvalues $\lambda_1 = 1$ and $\lambda_2 = 2$ both eigenspaces have dimension 1 with directions $[1, -2, -20]^T$ and $[0, 0, 1]^T$, respectively.

$$\text{Let } \mathbf{P} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & -20 & 1 \end{bmatrix}. \text{ Then } \mathbf{P}^{-1} \mathbf{A} \mathbf{P} = \mathbf{T} \text{ with } \mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 44 & 0 & 2 \end{bmatrix}.$$

3. When studying the system $\mathbf{z}'(t) = \mathbf{T}\mathbf{z}(t)$ row by row, we obtain

- $z'_1(t) = z_1(t)$ with solutions of the form $z_1(t) = \lambda e^t$, where $\lambda \in \mathbb{R}$.
- $z'_2(t) = z_2(t) + 2z_1(t)$. The solutions of the homogeneous equation $z'_2(t) = z_2(t)$ are $z_2(t) = \mu e^t$ with $\mu \in \mathbb{R}$. Since $z_1(t) = \lambda e^t$, a particular solution is $2\lambda t e^t$. Thus, we obtain the general solution: $z_2(t) = (2\lambda t + \mu)e^t$.
- $z'_3(t) = 2z_3(t) + 44z_1(t)$ is solved similarly and we obtain $z_3(t) = \nu e^{2t} - 44\lambda e^t$.

The result can be summarized in $\mathbf{z}(t) = \begin{bmatrix} \lambda e^t \\ (2\lambda t + \mu)e^t \\ \nu e^{2t} - 44\lambda e^t \end{bmatrix}$ and we deduce that the solution of the homogeneous system $\mathbf{y}'(t) = \mathbf{A}\mathbf{y}(t)$ has the form $\mathbf{y}(t) = \mathbf{P}\mathbf{z}(t) = \begin{bmatrix} (2\lambda t + \lambda + \mu)e^t \\ -2(2\lambda t + \mu)e^t \\ \nu e^{2t} - (40\lambda t + 20\mu + 44\lambda)e^t \end{bmatrix}$ with parameters $\lambda, \mu, \nu \in \mathbb{R}$.

4. $\mathbf{y}(t) = \alpha t + \beta$ with $\alpha = [\alpha_1, \alpha_2, \alpha_3]^T$ and $\beta = [\beta_1, \beta_2, \beta_3]^T$ is a solution of (12.9) if and only if

$$\begin{cases} \alpha_1 &= 3(t\alpha_1 + \beta_1) + t\alpha_2 + \beta_2 - 2t - 4 \\ \alpha_2 &= -4(t\alpha_1 + \beta_1) - (t\alpha_2 + \beta_2) + 3t + 6 \\ \alpha_3 &= 4(t\alpha_1 + \beta_1) - 8(t\alpha_2 + \beta_2) + 2(t\alpha_3 + \beta_3) - 16t - 16 \end{cases}$$

for any $t \in \mathbb{R}$ which is equivalent to the six equations

$$\begin{aligned} 3\alpha_1 + \alpha_2 &= 2, & 4\alpha_1 + \alpha_2 &= 3, & \alpha_3 &= -2\alpha_1 + 4\alpha_2 + 8, \\ 3\beta_1 + \beta_2 &= \alpha_1 + 4, & -4\beta_1 - \beta_2 &= \alpha_2 - 6, & 2\beta_3 &= \alpha_3 - 4\beta_1 + 8\beta_2 + 16. \end{aligned}$$

with unique solution $\alpha = [1, -1, 2]^T$ and $\beta = [2, -1, 1]^T$. A particular solution of (12.9) is then $\mathbf{y}(t) = \begin{bmatrix} t+2 \\ -t-1 \\ 2t+1 \end{bmatrix}$.

5. Combining the two previous questions, the general solution of (12.9) is $\mathbf{y}(t) = \begin{bmatrix} (2\lambda t + \lambda + \mu)e^t + t + 2 \\ -2(2\lambda t + \mu)e^t - t - 1 \\ \nu e^{2t} - (40\lambda t + 20\mu + 44\lambda)e^t + 2t + 1 \end{bmatrix}$. With the initial conditions (12.10) we deduce that $\lambda = 0$, $\mu = -2$ and $\nu = -40$ so that the unique solution of (12.9)–(12.10) is $\mathbf{y}(t) = \begin{bmatrix} -2e^t + t + 2 \\ -4e^t - t - 1 \\ 40(e^t - e^{2t}) + 2t + 1 \end{bmatrix}$.

Solutions of Exercise 12.3

1. f2.m

```
function v=f2(t,u);
v=[3 1 0;-4 -1 0;4 -8 2]*u+[-2*t-4;3*t+6;-16*t-16];
```

2. eulermeth.m

```
function [t,u]=eulermeth(namefunc,I,eta,n)
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)=eta;
for i=1:n
    u(:,i+1)=u(:,i)+h*feval(namefunc,t(i),u(:,i));
end
```

3. phi3.m

```
function v=phi3(t,u,h)
v=(inv(eye(3)-h*[3 1 0;-4 -1 0;4 -8 2])*...
(u+h*[-2*(t+h)-4;3*(t+h)+6;-16*(t+h)-16])-u)/h;
```

4. backeulermeth.m

```
function [t,u]=backeulermeth(namefunction,I,eta,n)
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)=eta;
for i=1:n
    u(:,i+1)=u(:,i)+h*feval(namefunction,t(i),u(:,i),h);
end
```

5. lineardiffssystem.m

```
n=200
eta=[0;3;1];
namefunction='f2';
[t,u]=eulermeth(namefunction,[0,1],eta,n);
subplot(1,2,1)
plot(t,u(1,:),t,u(2,:),t,u(3,:))
title('Euler''s method')

namefunction='phi3';
[t,v]=backwardeulermeth(namefunction,[0,1],eta,n);
subplot(1,2,2)
plot(t,v(1,:),t,v(2,:),t,v(3,:))
title('Backward Euler''s method')
```

6. errorstudy.m

```
arr_n=100:100:1000;
eta=[0;3;1];
namefunction='f2';
for i=1:length(arr_n)
    n=arr_n(i);
```

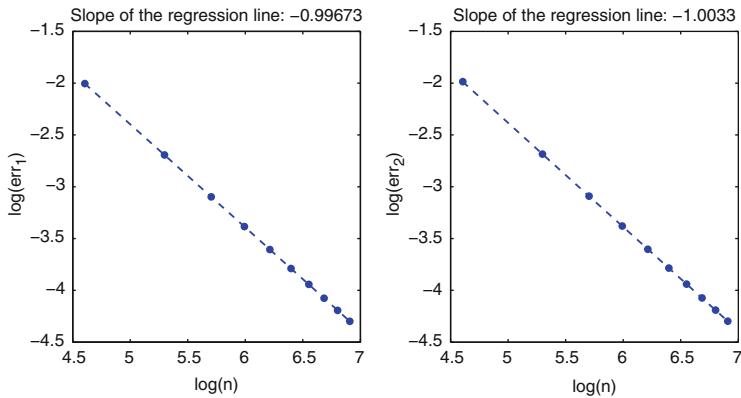


Fig. 12.10 Errors. *Left:* Euler's method, *Right:* Backward Euler method

```
[t,u]=eulermeth('f2',[0,1],eta,n);
[t,v]=backwardeulermeth('phi3',[0,1],eta,n);
y1=-2*exp(t)+t+2;
y2=4*exp(t)-t-1;
y3=-40*exp(2*t)+40*exp(t)+2*t+1;
arr_error1(i)=norm(y1-u(1,:),inf)...
+norm(y2-u(2,:),inf)+norm(y3-u(3,:),inf);
arr_error2(i)=norm(y1-v(1,:),inf)...
+norm(y2-v(2,:),inf)+norm(y3-v(3,:),inf));
end
subplot(1,2,1)
plot(log(arr_n),log(arr_error1),'--');
axis([4.5 7 -4.5 -1.5])
a=polyfit(log(arr_n),log(arr_error1),1);
title(['Slope of the regression line: ',...
num2str(a(1))])

subplot(1,2,2)
plot(log(arr_n),log(arr_error2),'--');
axis([4.5 7 -4.5 -1.5])
a=polyfit(log(arr_n),log(arr_error2),1);
title(['Slope of the regression line: ',...
num2str(a(1))])
```

The slopes in Fig. 12.10 give the orders which are both equal to one.

12.5.2 Other Examples of One Step Methods

Solutions of Exercise 12.4

Since f satisfies the Lipschitz condition, there exists a positive real number Λ such that for all $t \in [t_1, t_1 + T]$ and all $\mathbf{u}, \mathbf{v} \in \mathbb{R}^m$, $\|f(t, \mathbf{u}) - f(t, \mathbf{v})\| \leq \Lambda \|\mathbf{u} - \mathbf{v}\|$.

1. a. With the definition in (12.11)

$$\begin{aligned}
 \|\phi(t, \mathbf{u}, h) - \phi(t, \mathbf{v}, h)\| &= \frac{1}{2} \|f(t, \mathbf{u}) + f(t+h, \mathbf{u} + hf(t, \mathbf{u})) \\
 &\quad - f(t, \mathbf{v}) - f(t+h, \mathbf{v} + hf(t, \mathbf{v}))\| \\
 &\leq \frac{\Lambda}{2} (\|\mathbf{u} - \mathbf{v}\| + \|\mathbf{u} + hf(t, \mathbf{u}) - \mathbf{v} - hf(t, \mathbf{v})\|) \\
 &\leq \frac{\Lambda}{2} (2\|\mathbf{u} - \mathbf{v}\| + h\|f(t, \mathbf{u}) - f(t, \mathbf{v})\|) \\
 &\leq \frac{\Lambda}{2} (2 + h\Lambda) \|\mathbf{u} - \mathbf{v}\|.
 \end{aligned}$$

Thus From (12.6) we deduce that the numerical scheme is stable.

- b. For \mathbf{y} a regular solution of (12.1) i.e. such that $\mathbf{y}'(t) = f(t, \mathbf{y}(t))$ for any $t \in [t_1, t_1 + T]$, we have

$$\begin{aligned}
 \delta^h(\mathbf{y}) &:= \|\mathbf{y}(t+h) - \mathbf{y}(t) - h\phi(t, \mathbf{y}(t), h)\| \\
 &= \|\mathbf{y}(t+h) - \mathbf{y}(t) \\
 &\quad - \frac{h}{2} (f(t, \mathbf{y}(t)) + f(t+h, \mathbf{y}(t) + hf(t, \mathbf{y}(t))))\|
 \end{aligned}$$

Now

$$\begin{aligned}
 &f(t, \mathbf{y}(t)) + f(t+h, \mathbf{y}(t) + hf(t, \mathbf{y}(t))) \\
 &= \mathbf{y}'(t) + \mathbf{y}'(t+h) - f(t+h, \mathbf{y}(t+h)) \\
 &\quad + f(t+h, \mathbf{y}(t) + hf(t, \mathbf{y}(t))).
 \end{aligned}$$

Hence

$$\begin{aligned}
 \delta^h(\mathbf{y}) &\leq \|\mathbf{y}(t+h) - \mathbf{y}(t) - \frac{h}{2} (\mathbf{y}'(t) + \mathbf{y}'(t+h))\| \\
 &\quad + \frac{h}{2} \|-f(t+h, \mathbf{y}(t+h)) + f(t+h, \mathbf{y}(t) + hf(t, \mathbf{y}(t)))\| \\
 &\leq \|\mathbf{y}(t+h) - \mathbf{y}(t) - \frac{h}{2} (\mathbf{y}'(t) + \mathbf{y}'(t+h))\| \\
 &\quad + \frac{h}{2} \Lambda \|\mathbf{y}(t+h) - \mathbf{y}(t) - hf(t, \mathbf{y}(t))\| \\
 &\leq \|\mathbf{y}(t+h) - \mathbf{y}(t) - \frac{h}{2} (\mathbf{y}'(t) + \mathbf{y}'(t+h))\| \\
 &\quad + \frac{h}{2} \Lambda \|\mathbf{y}(t+h) - \mathbf{y}(t) - h\mathbf{y}'(t)\|
 \end{aligned}$$

Let $M_i = \max_{t \in [t_1, t_1 + T]} \|\mathbf{y}^{(i)}(t)\|$.

- Consider bounding $\|\mathbf{y}(t+h) - \mathbf{y}(t) - \frac{h}{2} (\mathbf{y}'(t) + \mathbf{y}'(t+h))\|$. We use Taylor expansions of $\mathbf{y}(t) \in C^3(\mathbb{R}, \mathbb{R}^m)$. Since m can be larger than 1,

$$\begin{aligned}\|\mathbf{y}(t+h) - \mathbf{y}(t) + h\mathbf{y}'(t) - \frac{h^2}{2}\mathbf{y}''(t)\| &\leq \frac{h^3}{6}M_3 \\ \|\mathbf{y}'(t+h) - \mathbf{y}'(t) - h\mathbf{y}''(t)\| &\leq \frac{h^2}{2}M_3\end{aligned}$$

so that

$$\begin{aligned}&\left\|\mathbf{y}(t+h) - \mathbf{y}(t) - \frac{h}{2}(\mathbf{y}'(t)) + \mathbf{y}'(t+h)\right\| \\ &= \left\|\mathbf{y}(t+h) - \mathbf{y}(t) - h\mathbf{y}'(t) - \frac{h^2}{2}\mathbf{y}''(t)\right. \\ &\quad \left.- \frac{h}{2}(\mathbf{y}'(t+h) - \mathbf{y}'(t) - h\mathbf{y}''(t))\right\| \\ &\leq \left\|\mathbf{y}(t+h) - \mathbf{y}(t) - h\mathbf{y}'(t) - \frac{h^2}{2}\mathbf{y}''(t)\right\| \\ &\quad + \frac{h}{2}\left\|\mathbf{y}'(t+h) - \mathbf{y}'(t) - h\mathbf{y}''(t)\right\| \\ &\leq h^3\left(\frac{1}{6} + \frac{1}{4}\right)M_3 = h^3\frac{5M_3}{12}\end{aligned}$$

$$\bullet \quad \frac{h}{2}\Lambda\|\mathbf{y}(t+h) - \mathbf{y}(t) - h\mathbf{y}'(t)\| \leq \frac{h}{2}\Lambda\frac{h^2M_2}{2} = h^3\frac{\Lambda M_2}{4}.$$

We deduce that $\delta^h(\mathbf{y}) = O(h^3)$.

- c. The consistency error $\varepsilon(\mathbf{y}) := \sum_{i=1}^n \delta_i^h(\mathbf{y})$ satisfies $\varepsilon(\mathbf{y}) \leq C_1 \sum_{i=1}^n H^3$.

Since $nH \leq T$, we deduce that $\varepsilon(\mathbf{y}) \leq C_2 H^2$. Then the order of the scheme is 2 and with the stability, we obtain convergence.

2. By (12.6) it is sufficient to show that there exist positive constants H^* , Λ such that for all $t \in I$, $\mathbf{u}, \mathbf{v} \in E$ and $h \in [0, H^*]$

$$\|\psi(t, \mathbf{u}, h) - \psi(t, \mathbf{v}, h)\| \leq \Lambda\|\mathbf{u} - \mathbf{v}\|. \quad (12.31)$$

Fix $t \in I$ and $\mathbf{u}, \mathbf{v} \in E$. For $h > 0$ we have

$$\mathbf{w} = \mathbf{u} + hf\left(t + \frac{h}{2}, \frac{1}{3}\mathbf{u} + \frac{2}{3}\mathbf{w}\right), \quad \mathbf{x} = \mathbf{v} + hf\left(t + \frac{h}{2}, \frac{1}{3}\mathbf{u} + \frac{2}{3}\mathbf{x}\right),$$

where $\mathbf{w} := \psi(t, \mathbf{u}, h)$ and $\mathbf{x} := \psi(t, \mathbf{v}, h)$. With the Lipschitz constant L of f , we deduce that

$$\begin{aligned}\|\mathbf{w} - \mathbf{x}\| &= \|\mathbf{u} - \mathbf{v} + h[f\left(t + \frac{h}{2}, \frac{1}{3}\mathbf{u} + \frac{2}{3}\mathbf{w}\right) - f\left(t + \frac{h}{2}, \frac{1}{3}\mathbf{v} + \frac{2}{3}\mathbf{x}\right)]\| \\ &\leq \|\mathbf{u} - \mathbf{v}\| + hL\left\|\frac{1}{3}\mathbf{u} + \frac{2}{3}\mathbf{w} - \frac{1}{3}\mathbf{v} - \frac{2}{3}\mathbf{x}\right\| \\ &\leq \|\mathbf{u} - \mathbf{v}\| + h\frac{L}{3}\|\mathbf{u} - \mathbf{v}\| + h\frac{2L}{3}\|\mathbf{w} - \mathbf{x}\|,\end{aligned}$$

from which it follows that

$$(1 - h \frac{2L}{3}) \|\mathbf{w} - \mathbf{x}\| \leq (1 + h \frac{L}{3}) \|\mathbf{u} - \mathbf{v}\|.$$

For $0 < h \leq H^* < \frac{3}{2L}$ we obtain

$$\|\psi(t, \mathbf{u}, h) - \psi(t, \mathbf{v}, h)\| = \|\mathbf{w} - \mathbf{x}\| \leq A \|\mathbf{u} - \mathbf{v}\|, \quad A := \frac{1 + H^* \frac{L}{3}}{1 - H^* \frac{2L}{3}} > 0.$$

By (12.31) we conclude that the numerical scheme is stable.

Solutions of Exercise 12.5

1. f1.m

```
function v= f1 (t ,u)
v=-150*u+49-150*t ;
end
```

2. Heun.m

```
function [ t ,u]=Heun( namefunc ,I ,eta ,n)
h=(I(2)-I (1))/n;
t=I (1):h:I (2);
u (:,1)= eta ;
for i=1:n
    x=feval(namefunc ,t (i ),u (:, i ));
    u (:, i +1)=u (:, i )+h/2*(x+...
        feval(namefunc ,t (i )+h,u (:, i )+h*x));
end
```

3. Heungraph.m

```
n=73
epsilon=0.01;
[t ,u]=Heun('f1',[0 ,1],1/3+epsilon ,n);
y=epsilon*exp(-150*t)-t+1/3;
plot(t,u,'--',t,y)
error=norm(u-y ,inf)
```

4. Heunerror.m

```
arrn=5000:10:5100;
epsilon=0.01;
for i=1:length (arrn)
    n=arrn (i );
    [t ,u]=Heun('f1',[0 ,1],1/3+epsilon ,n);
    y=epsilon*exp(-150*t)-t+1/3;
    arrerror (i )=norm(u-y ,inf );
end
plot(log(arrn ),log(arrerror ),'--')
```

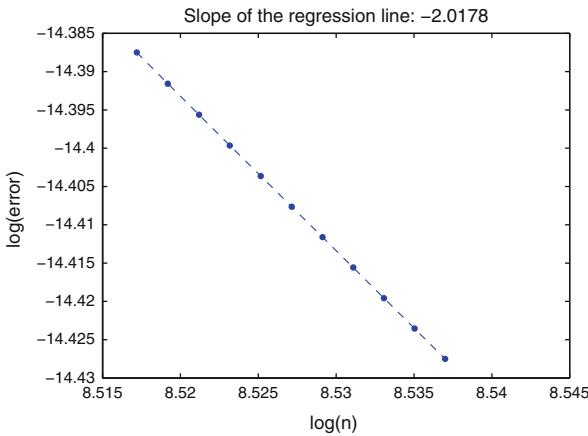


Fig. 12.11 Error in the Heun's Method for n between 5000 and 5100, step 100

```
a=polyfit(log(arrn),log(arreerror),1);
title(['Slope of the regression line: ',...
    num2str(a(1))])
xlabel('log(n)')
ylabel('log(error)')
```

See the graph in Fig. 12.11

Comment: The order of the method is 2 and we must use large values of n to reach a good approximation with this order. ☺

Solution of Exercises 12.6

1. Let $E(t) := T(t) + V(t)$. Then

$$\begin{aligned} E'(t) &= z(t)z'(t) + kx(t)x'(t) - x'(t)x(t)^3 \\ &= z(t)(-kx(t) + x(t)^3) + kx(t)z(t) - z(t)x(t)^3 = 0 \end{aligned}$$

so that $E(t)$ is constant.

2. If $f2(t, \mathbf{u}) := [u_2, -ku_1 + u_1^3]^T$, then $\mathbf{y}'(t) = f2(t, \mathbf{y}(t))$, where $\mathbf{y}(t) = [x(t), z(t)]^T$.
3. f2.m

```
function z=f2(t,u)
z(1,1)=u(2);
z(2,1)=-u(1)+u(1)^3;
```

4. springerror.m

```
arrn=5000:5010;
I=[0, 15];
eta=[0.2;0.5];
for i=1:length(arrn)
    n=arrn(i);
```

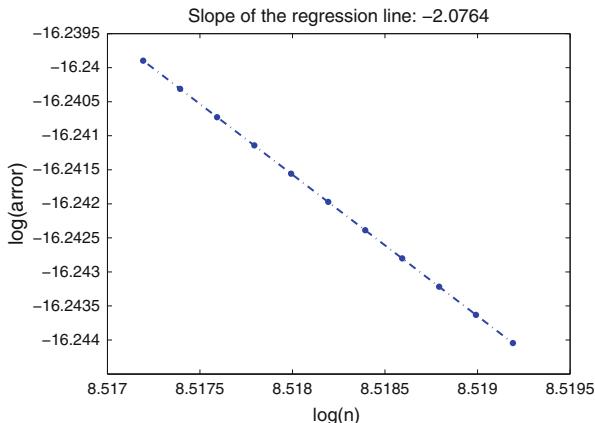


Fig. 12.12 The error between $E = V + T$ and its approximation

```

[ t ,u ]=Heun( 'f2' ,I ,eta ,n );
T=u( 2 ,:).^2/2;
V=u( 1 ,:).^2/2 - u( 1 ,:).^4/4;
E=T+V;
arrerror ( i )=norm(E-E( 1 ),inf);
end
plot ( log ( arrn ), log ( arrerror ), '.-.' );
a=polyfit ( log ( arrn ), log ( arrerror ), 1 );
title ([ ' Slope of the regression line: ' ,...
    num2str ( a( 1 ) ) ])
xlabel ( ' log ( n ) ' )
ylabel ( ' log ( arrror ) ' )

```

The graph is in Fig. 12.12 where we see that the numerical order is 2.

Solution of Exercise 12.7

1. If $f3(t, \mathbf{u}) = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \\ \frac{1}{2} \cos(2t) \sin t - u_3 u_1^2 \end{bmatrix}$ and $\boldsymbol{\eta} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ -1 \end{bmatrix}$, then (12.15) is satisfied.
2. f3.m

```

function z=f3 ( t ,u )
z=[u( 2 ),u( 3 ),u( 4 ),1/2 * sin ( 2 * t ) * cos ( t )-u( 3 ) * u( 1 )^2 ] ;

```

3. RK4.m.

```

function [ t ,u ]=RK4( namefunc ,I ,eta ,n )
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)= eta ;

```

```

for i=1:n
    u1=u(:, i);
    k1=feval(namefunc , t(i) , u1);
    k2=feval(namefunc , t(i)+h/2 , u1+h/2*k1);
    k3=feval(namefunc , t(i)+h/2 , u1+h/2*k2);
    k4=feval(namefunc , t(i)+h , u1+h*k3);
    u(:, i+1)=u(:, i)+h/6*(k1+2*k2+2*k3+k4);
end

```

4. order4graph.m.

```

n=15;
I=[0,2*pi];
eta=[0,1,0,-1]';
[t,u]=Heun('f3',I,eta,n);
[t,v]=RK4('f3',I,eta,n);
plot(t,sin(t),t,u(1,:),'--',t,v(1,:),'--')

```

5. RK4error.m

```

arrn =[100:10:200];
I=[0,2*pi];
eta=[0,1,0,-1]';
for i=1:length(arrn)
    [t,u]=RK4('f3',I,eta,arrn(i));
    arrerror(i)=norm(sin(t)-u(1,:),inf);
end
plot(log(arrn),log(arrerror),'-')
a=polyfit(log(arrn),log(arrerror),1)
title(['Slope of the regression line: ',...
    num2str(a(1))])
xlabel('log(n)')
ylabel('log(error)')

```

The graph is in Fig. 12.13 where the numerical order of the method is 4.

Solution of Exercise 12.8

- On J the set of solutions of $(1-t^2)y'(t) + (2t-1)y(t) = 0$ is a vector space of dimension 1. Since $\frac{2t-1}{t^2-1} = \frac{1/2}{t-1} + \frac{3/2}{t+1}$, if $y(t)$ is a non vanishing solution then $\frac{y'(t)}{y(t)} = \frac{2t-1}{t^2-1}$ can be integrated giving

$$\ln|y(t)| = 1/2 \ln|t-1| + 3/2 \ln|t+1| + c, \quad c \in \mathbb{R}$$

from which we deduce that $y(t) = \lambda|t-1|^{1/2}|t+1|^{3/2}$ on J .

- Looking for a solution of $(1-t^2)y'(t) + (2t-1)y(t) = 1$ in the form $y(t) = at^2 + bt + c$, we find $y(t) = (2t^2 + 2t - 1)/3$ on any interval $J \subset \mathbb{R}$.

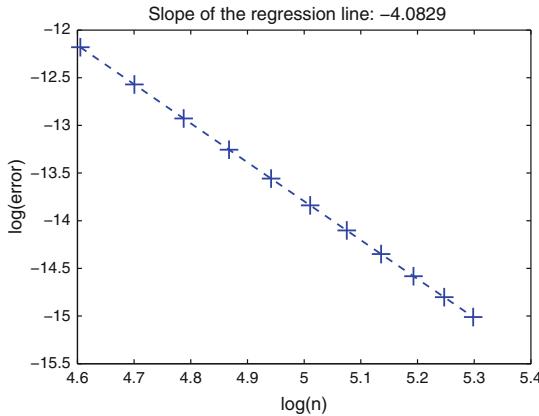


Fig. 12.13 Error between the exact solution and the approximation by Runge Kutta's method

3. The general solution of the differential equation is given by

$y(t) = \lambda|t - 1|^{1/2}|t + 1|^{3/2} + (2t^2 + 2t - 1)/3$ on any interval J such that $J \cap \{-1, 1\} = \emptyset$. Clearly the function can be extended by continuity at $t = 1$ by $y(1) := 1$ and at $t = -1$ by $y(-1) := -1/3$.

With these extensions $y(t) = \lambda|t - 1|^{1/2}|t + 1|^{3/2} + (2t^2 + 2t - 1)/3$ is a solution on any interval $I \subset \mathbb{R}$. If we add the initial condition $y(0) = 2/3$ then the solution of (12.16) is

$$y(t) = |t - 1|^{1/2}|t + 1|^{3/2} + (2t^2 + 2t - 1)/3, \quad t \in I.$$

4. f4.m

```
function v=f4(t,u)
v=(1-(2*t-1)*u)/(1-t^2);
```

5. See solution of Exercise 12.7 for RK4.m.

6. extendgraph.m

```
n=20;
I=[0,pi/2]
eta=2/3;
[t,u]=RK4('f4',I,eta,n);
solex= abs(t-1).^(1/2).*abs(t+1).^(3/2)...
+(2*t.^2+2*t-1)/3;
plot(t,solex,t,u,'--')
```

The graphs are shown in Fig. 12.14 where we see that it is impossible to have an extension beyond 1 by this method. Let us notice that MATLAB does not indicate an error and, the programmer must analyze the function *f4*.

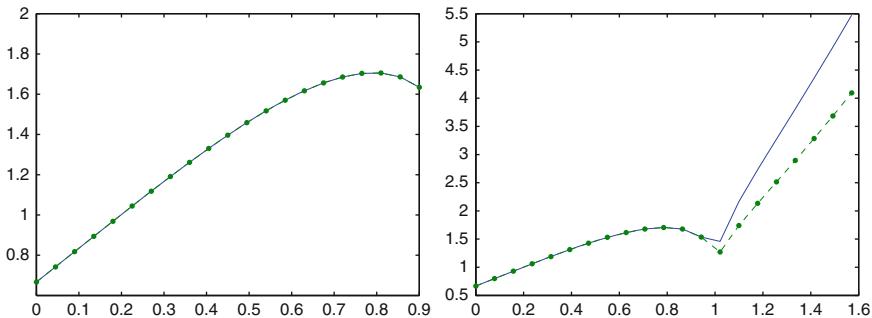


Fig. 12.14 Exact solution and the approximation by Runge Kutta's method on $[0, 0.9]$ on the *left* and $[0, \pi/2]$ on the *right*

12.5.3 Predictor-Corrector

Solution of Exercise 12.9

1. f3.m

```
function z=f3(t,u)
z=[u(2),u(3),u(4),1/2*sin(2*t)*cos(t)-u(3)*u(1)^2]';
```

2. prediccorrect1.m

```
function [t,u]=prediccorrect1(namefunc,I,eta,n)
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)=eta;
for i=1:n
    k=feval(namefunc,t(i),u(:,i));
    v=u(:,i)+h*k;
    u(:,i+1)=u(:,i)+h/2*(k+feval(namefunc,t(i+1),v));
endt
```

prediccorrect2.m

```
function [t,u]=prediccorrect2(namefunc,I,eta,n)
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)=eta;
for i=1:n
    k=feval(namefunc,t(i),u(:,i));
    v=u(:,i)+h*k;
    w=u(:,i)+h/2*(k+feval(namefunc,t(i+1),v));
    u(:,i+1)=u(:,i)+h/2*(k+feval(namefunc,t(i+1),w));
end
```

3. pcgraph.m

```
n=15;
[t,u]=predictcorrect1('f3',[0,2*pi],[0,1,0,-1]',15)
[t,v]=predictcorrect2('f3',[0,2*pi],[0,1,0,-1]',15)
plot(t,sin(t),t,u(1,:),'-.',t,v(1,:),'+-')
```

4. pcerror.m

```
arrn=1000:10:1100;
for i=1:length(arrn)
    n=arrn(i);
    [t,u]=predictcorrect1('f3',[0,2*pi],[0,1,0,-1]',n);
    [t,v]=predictcorrect2('f3',[0,2*pi],[0,1,0,-1]',n);
    exsol=sin(t);
    arrrerror1(i)=norm(u(1,:)-exsol,inf);
    arrrerror2(i)=norm(v(1,:)-exsol,inf);
end
subplot(1,2,1)
plot(log(arrn),log(arrrerror1),'o--');
a=polyfit(log(arrn),log(arrrerror1),1);
title(['Slope: ',num2str(a(1))])
xlabel('log(n)')
ylabel('log(error)')
subplot(1,2,2)
plot(log(arrn),log(arrrerror2),'+-');
a=polyfit(log(arrn),log(arrrerror2),1);
title(['Slope: ',num2str(a(1))])
xlabel('log(n)')
ylabel('log(error)')
```

The graph is in Fig. 12.15. In both cases, the numerical order is 2 and the second method with graph on the right gives a smaller error $\approx 3.5 \times 10^{-5}$ instead of $\approx 4 \times 10^{-4}$.

12.5.4 Application of ODE Solvers to the Shooting Method

Solution of Exercise 12.10

1. $\lambda = \frac{a - y_{[1]}(1)}{y_{[0]}(1) - y_{[1]}(1)}$.
2. If $f(t, u) = \begin{bmatrix} u_2 \\ 1 - 2t \cos t - u_1 + tu_2 \end{bmatrix}$, then (\mathcal{CP}_s) is transformed into (\mathcal{DS}_s) .
3. a. f.m

```
function v=f(t,u)
v=[u(2);1-2*t*cos(t)-u(1)+t*u(2)];
```

- b. eulermeth.m is given in the solution of Exercise 12.3
- c. shooteuler.m

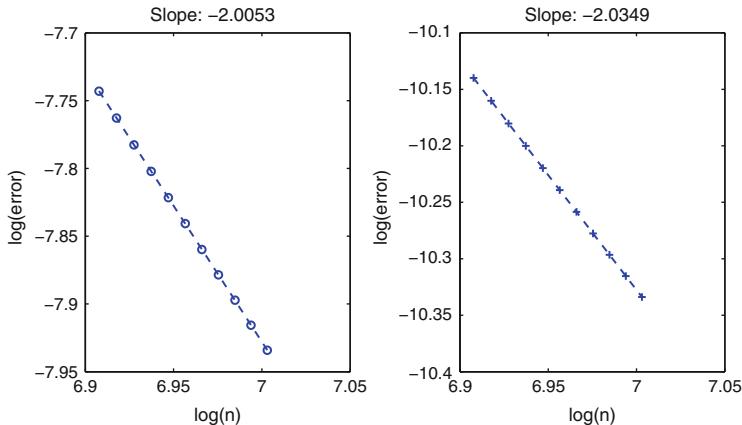


Fig. 12.15 Error using predictor–corrector methods, Euler+Trapezoidal on the *left* and Euler+Twice Trapezoidal on the *right*

```

function [t ,u]=shooteuler(namefunction ,n ,a)
[t ,U0]=eulermeth(namefunction ,[0 ,1],[1;0] ,n);
[t ,U1]=eulermeth(namefunction ,[0 ,1],[1;1] ,n);
l=(a-U1(1 ,n+1))/(U0(1 ,n+1)-U1(1 ,n+1));
u=1*U0(1 ,:)+(1-l)*U1(1 ,:);
```

d. `errorshooteuler1.m.`

```

n=4;
a=2*(1+sin(1));
[t ,u]=shooteuler('f' ,n ,a);
plot(t ,u ,'.--',t ,t+2*sin(t)+1)
error=norm(u-t-2*sin(t)-1,inf)
```

e. `errorshooteuler2.m.`

```

arrn =[5 ,10 ,20 ,50 ,100 ,200];
a=2*(1+sin(1));
for i=1:length(arrn)
    n=arrn(i);
    [t ,u]=shooteuler('f' ,n ,a);
    arrerror(i)=norm(u-t-2*sin(t)-1,inf);
end
plot(log(arrn),log(arrerror),'o--');
xlabel('log(n)');
ylabel('log(error)');
a=polyfit(log(arrn),log(arrerror),1)
title([' Slope of the regression line: ', ...
    num2str(a(1))]);
```

See next question for the graph and comment.

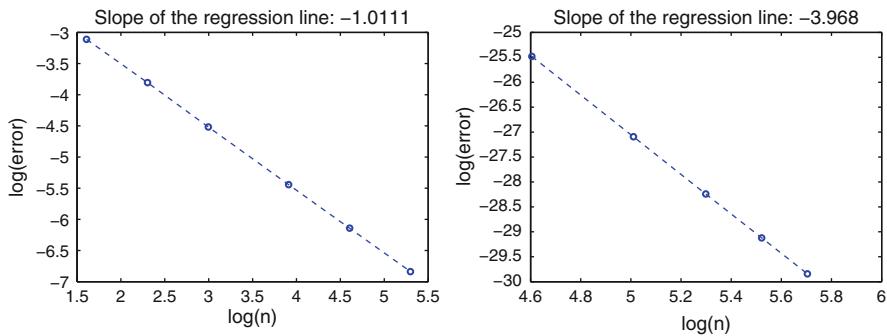


Fig. 12.16 Error between exact and discrete solutions with Euler's method on the *left* and Runge–Kutta's method on the *right*

- f. The program RK4.m is given in the solution of Exercise 12.7.

shootRK.m, errorshootRK1.m, errorshootRK2.m are similar to the programs with Euler's method. For the last program, we begin with $arrn = [100 : 50 : 300]$.

In Fig. 12.16, the scales are different. The order is 1 with Euler's method and 4 with the Runge–Kutta method.

Solution of Exercise 12.11

1. $f(t, u) = [u_2, -u_1]^T$
f.m

```
function v=f(t,u)
v=[u(2); -u(1)];
```

2. RK3.m

```
function [t,u]=RK3(namefunc,I,eta,n)
h=(I(2)-I(1))/n;
t=I(1):h:I(2);
u(:,1)=eta;
for i=1:n
    u(:,i+1)=u(:,i)+h/6*(k1+4*k2+k3);
    k1=feval(namefunc,t(i),u1);
    k2=feval(namefunc,t(i)+h/2,u1+h/2*k1);
    k3=feval(namefunc,t(i)+h,u1-h*k1+2*h*k2);
end
```

3. RK3error1.m

```
n=4;
[t,u]=RK3('f',[0,pi],[0;1],n);
tt=0:pi/500:pi;
plot(t,sin(t),'b.',tt,sin(tt),t,u(1,:),'x-r')
err=norm(sin(t)-u(1,:),inf)
```

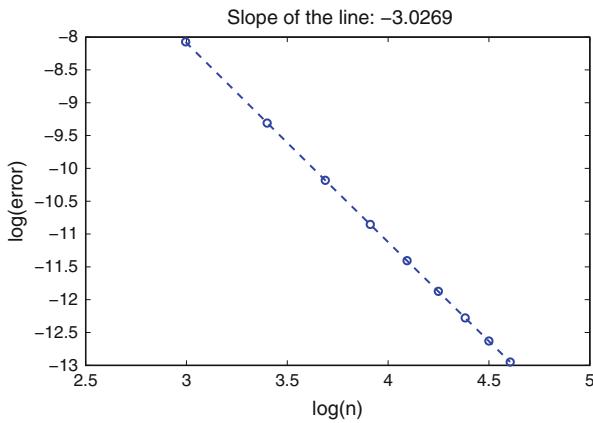


Fig. 12.17 Exact and discrete solution by RK3 method

4. RK3error2.m

```
arrn=20:10:100;
for i=1:length(arrn)
    n=arrn(i);
    [t,u]=RK3('f',[0,pi],[0;1],n);
    err(i)=norm(sin(t)-u(1,:),inf);
end
plot(log(arrn),log(err),'o--');
a=polyfit(log(arrn),log(err),1);
xlabel('log(n)');
ylabel('log(error)');
title(['Slope of the line: ',num2str(a(1))])
```

The graph in Fig. 12.17 indicates order 3.

Solutions of Exercise 12.12

1. f.m

```
function v=f(t,u)
v=[u(2);3/2*u(1)^2];
```

2. w.m

```
function z=w(s,n)
[t,Us]=RK3('f',[0,1],[4;s],n);
z=Us(1,n+1)-1;
```

3. plotW.m

```
n=20;
s=-100:-1;
for i=1:length(s)
    z(i)=w(s(i),n);
end
```

```

plot(s,z,s,zeros(1,length(s)),'—')
title('Function w')
xlabel('s')
grid on

```

4. In Fig. 12.9, we see that the two roots of w are in $[-40, -30]$ and $[-10, 0]$.

5.6.7. Approxsol.m

```

n=20;
h=0.1;
n0=5;

s1=-36;
w1=w(s1,n);
s2=s1-w1*h/(w(s1+h,n)-w1);
for i=2:n0-1
    w2=w(s2,n);
    s=s2-w2*(s2-s1)/(w2-w1);
    s1=s2;
    w1=w2;
    s2=s;
end
sigma1=s2

s1=-7;
w1=w(s1,n);
s2=s1-w1*h/(w(s1+h,n)-w1);
for i=2:n0-1
    w2=w(s2,n);
    s=s2-w2*(s2-s1)/(w2-w1);
    s1=s2;
    w1=w2;
    s2=s;
end
sigma2=s2

[t ,Us1]=RK3('f',[0,1],[4; sigma1],n);
[t ,Us2]=RK3('f',[0,1],[4; sigma2],n);
plot(t ,Us1(1,:),'x—',t ,Us2(1,:),'+-.')

```

See Fig. 12.18.

8. We multiply $y''_{[s]}(t) - \frac{3}{2}y^2_{[s]}(t) = 0$ by $y'_{[s]}(t)$ and integrate to obtain

$(y')^2_{[s]}(\cdot)/2 - 3/2y^3_{[s]}(\cdot)/3 - (y')^2_{[s]}(0)/2 + 3/2y^3_{[s]}(0)/3 = 0$. With the initial conditions $y_{[s]}(0) = 4$ and $y'_{[s]}(0) = s$, we deduce that $(y')^2_{[s]} - y^3_{[s]} - s^2 + 64 = 0$ for any $s \in \mathbb{R}$,

9. error1.m

```

n=100;
h=0.1;
n0=5;
s1=-36;
w1=w(s1,n);
s2=s1-w1*h/(w(s1+h,n)-w1);

```

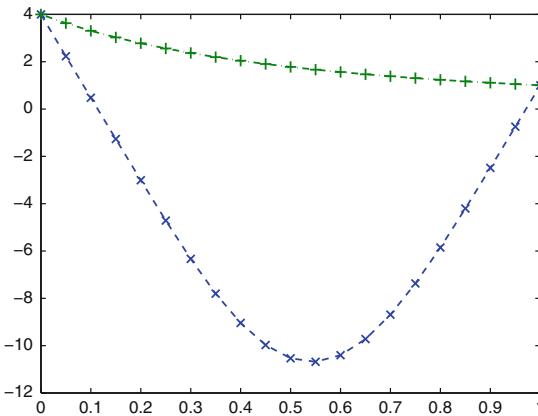


Fig. 12.18 Graph of the two approximations of the solutions of (SP) in (12.25)

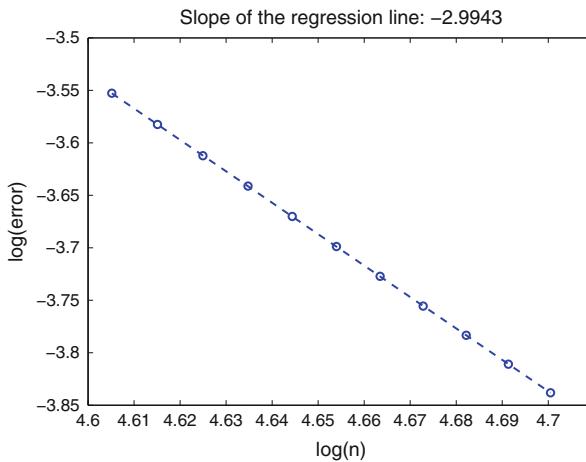


Fig. 12.19 Error function of n for the approximation of one solution of (SP) in (12.25)

```

for i=2:n0-1
    w2=w(s2 ,n );
    s=s2-w2*(s2-s1 )/(w2-w1 );
    s1=s2 ;w1=w2; s2=s ;
end
sigma1=s2
[t ,Us]=RK3( 'f ' ,[0 ,1 ],[4; sigma1 ] ,n );
err=norm(Us(2 ,:).^2 -Us(1 ,:).^3 -sigma1 ^2+64 ,inf )

```

10. error2.m

```

arrn=100:110;
h=0.1;
n0=5;
for j=1:length(arrn)
    n=arrn(j);
    s1=-36;
    w1=w(s1,n);
    s2=s1-w1*h/(w(s1+h,n)-w1);
    for i=2:n0-1
        w2=w(s2,n);
        s=s2-w2*(s2-s1)/(w2-w1);
        s1=s2;w1=w2;s2=s;
    end
    sigma1=s2;
    [t,Us]=RK3('f',[0,1],[4;sigma1],n);
    arrerr(j)=norm(Us(2,:).^2-Us(1,:).^3-sigma1.^2+64,inf);
end
plot(log(arrn),log(arrerr),'o—')
xlabel('log(n)');
ylabel('log(error)');
a=polyfit(log(arrn),log(arrerr),1)
title(['Slope of the regression line: ',num2str(a(1))])

```

In Fig. 12.19, we see that the computed order of the method is 3.

Chapter 13

Finite Differences for Differential and Partial Differential Equations

Finite differences are used to approximate derivatives of a function f , in order to solve differential and partial differential equations. In this way the continuous problem can be replaced by a discrete one. In the first section we consider finite differences that lead to approximations of first and second derivatives. We consider the univariate case in Exercise 13.1, and the bivariate Laplacian in Exercise 13.2.

In the next section, we use some of the previous approximations to define univariate discrete problems. In Exercise 13.3, we study a linear differential equation with boundary conditions and its approximation to obtain a method of order 2. We propose to improve this order with a new approximation in Exercise 13.4. Finally, once the previous exercises are understood, in Exercise 13.5, we propose a convection-diffusion problem for which we do not give details for the definition of the approximate problem and the programs.

Then, in the next sections, we study bivariate partial differential equations. First we study an equation involving the Laplacian in Exercise 13.6, then its numerical solution in Exercise 13.7. We extend the method to a nonlinear problem in Exercise 13.8. Finally, in Exercises 13.9 and 13.10, we study the heat equation using three numerical methods and estimate the order of approximation of the methods.

13.1 Definitions of Finite Differences

Exercise 13.1. Approximations of derivatives

Let $x, h \in \mathbb{R}$ with $h > 0$ and let f be a smooth enough function on an interval containing $x-h$ and $x+h$. Prove that there exist $\theta_1, \theta_2 \in (0, 1)$ and $\theta_3, \theta_4 \in (-1, 1)$ such that:

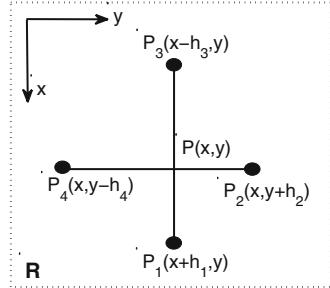


Fig. 13.1 The five points

$$\text{If } f \in C^2([x, x+h]), \quad f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2} f''(x + \theta_1 h) \quad (13.1)$$

$$\text{If } f \in C^2([x-h, x]), \quad f'(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2} f''(x - \theta_2 h) \quad (13.2)$$

$$\text{If } f \in C^3([x-h, x+h]), \quad f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f^{(3)}(x + \theta_3 h) \quad (13.3)$$

$$\text{If } f \in C^4([x-h, x+h]), \quad f''(x) = \frac{f(x-h) - 2f(x) + f(x+h)}{h^2} - \frac{h^2}{12} f^{(4)}(x + \theta_4 h) \quad (13.4)$$

▷ *Math Hint*¹ ◁

Exercise 13.2. Approximation of the Laplacian $\Delta\phi$

Pierre-Simon, marquis de Laplace (1749–1827) was a French mathematician and astronomer who worked on the development of mathematical astronomy and statistics. He published the five-volume Mécanique Céleste (Celestial Mechanics) (1799–1825). Laplace is remembered as one of the greatest scientists of all time.



Let $P = (x, y) \in \mathbb{R}^2$. For $h_i > 0$, $i = 1, 2, 3, 4$, we define $P_1 = (x + h_1, y)$, $P_2 = (x, y + h_2)$, $P_3 = (x - h_3, y)$, $P_4 = (x, y - h_4)$, see Fig. 13.1.

Let ϕ be a smooth function defined on a rectangle $R \subset \mathbb{R}^2$ containing the P_i s. Recalling that $\Delta\phi := \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2}$, we define its approximation $\Delta_h\phi$

$$\begin{aligned} \Delta_h\phi(P) := & \frac{2}{h_1(h_1 + h_3)}\phi(P_1) + \frac{2}{h_2(h_2 + h_4)}\phi(P_2) \\ & + \frac{2}{h_3(h_1 + h_3)}\phi(P_3) + \frac{2}{h_4(h_2 + h_4)}\phi(P_4) \\ & - \left\{ \frac{2}{h_1h_3} + \frac{2}{h_2h_4} \right\} \phi(P). \end{aligned} \quad (13.5)$$

¹ Use Taylor expansions with remainder.

1. Prove that $|\Delta_h \phi(P) - \Delta\phi(P)| \leq c_1 M_3 h$ where $c_1 \leq 2/3$, $M_3 := \max_{(x,y) \in R} \left(\left| \frac{\partial^3 \phi}{\partial x^3} \right|, \left| \frac{\partial^3 \phi}{\partial y^3} \right| \right)$ and $h := \max h_i$.
2. If $h_1 = h_2 = h_3 = h_4 = h$, show that (13.5) is simplified into

$$\Delta_h \phi := \frac{\phi(P_1) + \phi(P_2) + \phi(P_3) + \phi(P_4) - 4\phi(P)}{h^2}, \quad (13.6)$$

and then bound $|\Delta_h \phi(P) - \Delta\phi(P)|$ using partial derivatives of order 4.

13.2 Applications of Finite Differences in Dimension 1

Exercise 13.3. Bending of a beam

A beam of length 1 is stretched at its two ends along its axis by a force P and subject to a transverse load f . The bending moment u is solution of the problem

$$(P) \begin{cases} -u''(x) + c(x)u(x) = f(x) & \text{for } 0 \leq x \leq 1 \\ u(0) = u(1) = 0, \end{cases} \quad (13.7)$$

where $c(x) := P/EI(x)$, E is Young's modulus of the material and $I(x)$ is the moment of inertia of the section at the point x .

Thomas Young (1773–1829) was an English scientist who worked on the fields of vision, mechanic, energy... and Egyptology. He made innovations in Egyptian hieroglyphs (specifically the Rosetta Stone) before Champollion.



To simplify, in the following, we assume that $c(x)$ is a constant $c > 0$.

For a positive integer n and $h := 1/(n+1)$, on $[0, 1]$, we use a uniform partition $x_i := (i-1)h, i = 1, \dots, n+2$. We are looking for an approximate solution $\bar{v}_h := [v_1, \dots, v_{n+2}]^T$ of the exact solution $\bar{u} := [u(x_1), \dots, u(x_{n+2})]^T$. The initial and final values are known $v_1 = u(x_1) = 0$ and $v_{n+2} = u(x_{n+2}) = 0$ so that it remains to find $v_h := [v_2, \dots, v_{n+1}]^T \in \mathbb{R}^n$.

1. Find the solution of (P) if $f \in C^0([0, 1])$, and conclude that it is unique.
2. Suppose the exact solution u is in $C^4([0; 1])$. At x_i we use (13.4) to replace $u''(x_i)$ by $\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} - \frac{h^2}{12} u^{(4)}(x_i + \theta_4 h)$ in (13.7), then we ignore the 4th derivative term and denote the resulting approximations by $v_j \approx u(x_j)$ for $j = 2, \dots, n+1$. Show that the corresponding equations are

$$-\frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + cv_i = f(x_i), \quad i = 2, \dots, n+1, \quad v_1 = v_{n+2} = 0.$$

3. This leads to a discrete problem (P_h) that can be written in the form

$$\mathbf{A}_h \mathbf{v}_h = \mathbf{f}_h \text{ where } \mathbf{A}_h \in \mathbb{R}^{n \times n}, \mathbf{f}_h := h^2 [f(x_2), \dots, f(x_{n+1})]^T, \quad (13.8)$$

$$\text{where } \mathbf{A}_h := \begin{bmatrix} 2 + ch^2 & -1 & 0 & \dots & 0 \\ -1 & 2 + ch^2 & -1 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & -1 & 2 + ch^2 & -1 \\ 0 & \dots & 0 & -1 & 2 + ch^2 \end{bmatrix}.$$

That (P_h) has a unique solution can be proved using that \mathbf{A}_h is strictly diagonally dominant (cf. Exercise 2.8) or with Gershgorin's circle Theorem 8.2. We propose an alternative method that will also evaluate the error $\mathbf{v} - \mathbf{u}$.

4. A vector $\mathbf{v} \in \mathbb{R}^N$ is *positive*, written $\mathbf{v} > \mathbf{0}$, (respect. *non negative*, $\mathbf{v} \geq \mathbf{0}$) if all its components are positive (respect. non-negative). We have similar definitions a *positive* (respect. *non-negative*) matrix \mathbf{M} .

- a. Show that $\mathbf{A}_h \mathbf{v} \geq \mathbf{0} \Rightarrow \mathbf{v} \geq \mathbf{0}$. \triangleright Math Hint²
- b. Deduce that \mathbf{A}_h is non singular and $\mathbf{A}_h^{-1} \geq \mathbf{0}$. Thus (P_h) has a unique solution.

5. **Error:** We suppose that $u \in C^4([0, 1])$.

If \mathbf{w} and $\boldsymbol{\theta}$ are two vectors in \mathbb{R}^n such that $\mathbf{A}_h \mathbf{w} = \boldsymbol{\theta}$, we define the function $\psi(x) := \frac{x(1-x)}{2h^2} \|\boldsymbol{\theta}\|_\infty$ and the vector $\boldsymbol{\Psi} := [\psi(x_2), \dots, \psi(x_{n+1})]^T$.

- a. Compute $\mathbf{A}_h \boldsymbol{\Psi}$, then prove that $\mathbf{A}_h \boldsymbol{\Psi} \geq \mathbf{A}_h \mathbf{w}$ and $\mathbf{A}_h \boldsymbol{\Psi} \geq \mathbf{A}_h(-\mathbf{w})$.
- b. Deduce that $\|\mathbf{w}\|_\infty \leq \|\boldsymbol{\Psi}\|_\infty \leq \frac{1}{8h^2} \|\boldsymbol{\theta}\|_\infty$.
- c. Show that $\mathbf{A}_h(\mathbf{u}_h - \mathbf{v}_h) = c_1 h^4 [u^{(4)}(x'_2), \dots, u^{(4)}(x'_{n+1})]^T$ for a constant c_1 .
- d. Bound $\|\mathbf{u}_h - \mathbf{v}_h\|_\infty$ using h and $M_4 = \max_{x \in [0, 1]} |u^{(4)}(x)|$.

Comment: The error in the approximation of u'' was of order 2 and we keep the same order after the solution of the system. A numerical computation would show that the condition number (cf. Chap. 4) of \mathbf{A}_h does not increase with n . ☺

Exercise 13.4. Numerov scheme

Boris Vasilievich Numerov (1891–1941) was a Russian astronomer who created algorithms and numerical methods. Member of the Academy of Sciences and Professor at the University of Leningrad, he was arrested in October 1936, and sentenced to 10 years hard labour accused of being a spy for Germany based on the fact that German astronomers had named an asteroid after him. He was executed along with other political prisoners in September 1941 but in 1957, his memory was rehabilitated.



² Consider row i of $\mathbf{A}_h \mathbf{v}$ where $v_i := \min_{j=1, \dots, n} v_j$ to show that $v_i \geq 0$.

We recall the problem of the bending of a beam studied in Exercise 13.3.

$$(P) : \begin{cases} -u''(x) + c(x)u(x) = f(x), & x \in [0, 1] \\ u(0) = \alpha, u(1) = \beta. \end{cases}$$

where c and f are two smooth enough functions with $c \geq 0$. We recall that (P) has a unique solution.

For a positive integer n and $h := \frac{1}{n+1}$, again we use the uniform partition $\boldsymbol{x} := [x_1, \dots, x_{n+2}]^T$ with $x_i := (i-1)h$ for $i = 1, \dots, n+2$.

1. Prove that for $\varphi \in C^6([x-h, x+h])$

$$\left| \frac{\varphi(x-h) - 2\varphi(x) + \varphi(x+h)}{h^2} - \frac{\varphi''(x-h) + 10\varphi''(x) + \varphi''(x+h)}{12} \right| \leq \gamma h^4 \max_{t \in [x-h, x+h]} |\varphi^{(6)}(t)| \quad (13.9)$$

for some constant γ .

2. Since in particular (P) holds for all x_i we have

$$\begin{aligned} & -\frac{u''_{i-1} + 10u''_i + u''_{i+1}}{12} + \frac{c_{i-1}u_{i-1} + 10c_iu_i + c_{i+1}u_{i+1}}{12} \\ &= \frac{f_{i-1} + 10f_i + f_{i+1}}{12}, \quad i = 2, \dots, n+1, \end{aligned}$$

where $u_j := u(x_j)$, $c_j := c(x_j)$ and $f_j := f(x_j)$ for all j .

Using (13.9) to replace the first term we obtain an $O(h^4)$ approximation to (P) . Let $\bar{\boldsymbol{v}} := [v_1, \dots, v_{n+2}]^T$ be the approximation of $\bar{\boldsymbol{u}} := [u_1, \dots, u_{n+2}]^T$ with end values: $v_1 = \alpha$ and $v_{n+2} = \beta$. Show that the discrete problem is

$$(P_h) : \mathbf{A}\boldsymbol{v} = \mathbf{b}$$

where

$$\mathbf{A} := \begin{bmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{bmatrix} + \frac{1}{12} h^2 \begin{bmatrix} 10c_2 & c_3 & & 0 \\ c_2 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n+1} \\ 0 & c_n & 10c_{n+1} & \end{bmatrix} \in \mathbb{R}^{n \times n} \quad (13.10)$$

and

$$\mathbf{b} := \frac{h^2}{12} \begin{bmatrix} f_1 + 10f_2 + f_3 \\ \vdots \\ \vdots \\ f_n + 10f_{n+1} + f_{n+2} \end{bmatrix} + \begin{bmatrix} (1 - \frac{h^2}{12} c_1)\alpha \\ 0 \\ \vdots \\ 0 \\ (1 - \frac{h^2}{12} c_{n+2})\beta \end{bmatrix} \in \mathbb{R}^n \quad (13.11)$$

Then $\bar{\mathbf{v}} = \begin{bmatrix} \alpha \\ \mathbf{v} \\ \beta \end{bmatrix}$.

3. **Programs:** Write a program that for a given integer n computes \mathbf{x} , $\bar{\mathbf{f}} := [f_1, \dots, f_{n+2}]^T$ and $\bar{\mathbf{c}} := [c_1 \dots c_{n+2}]^T$ in \mathbb{R}^{n+2} using two function files `f1.m` and `c1.m`. Save as `numerov1.m`. Test: $n = 2$, $f(x) = f1(x) := (\pi^2 + x) \sin(\pi(x + 1/2))$, $c(x) = c1(x) := x$.

```
>> numerov1
bf =
    9.8696
    5.1015
   -5.2681
  -10.8696
bc =
     0
    0.3333
    0.6667
    1.0000
```

4. Extend the previous program with the computation of $\mathbf{A} = \mathbf{A}_1 + \frac{h^2}{12} \mathbf{A}_2$, where the two matrices in $\mathbb{R}^{n \times n}$ are given by

$$\mathbf{A}_1 := \begin{bmatrix} 2 & -1 & & 0 \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -1 & 2 \end{bmatrix}, \quad \mathbf{A}_2 := \begin{bmatrix} 10c_2 & c_3 & & 0 \\ c_2 & 10c_3 & c_4 & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n+1} \\ 0 & & & c_n & 10c_{n+1} \end{bmatrix}.$$

Save as `numerov2.m` and test with $n = 3$.

```
>> numerov2
A =
    2.0130    -0.9974      0
   -0.9987    2.0260   -0.9961
     0    -0.9974    2.0391
```

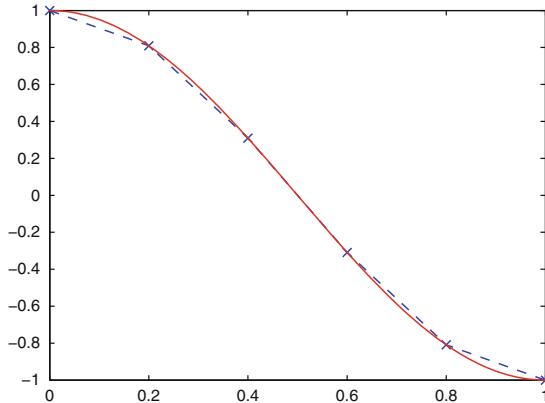


Fig. 13.2 Exact solution and Numerov approximation

5. Complete the previous program with the computation of $\mathbf{b} = \frac{h^2}{12} \mathbf{b}_1 + \mathbf{b}_2$, where $\mathbf{b}_1, \mathbf{b}_2 \in \mathbb{R}^n$ are given by

$$\mathbf{b}_1 := \begin{bmatrix} f_1 + 10f_2 + f_3 \\ \vdots \\ \vdots \\ f_n + 10f_{n+1} + f_{n+2} \end{bmatrix}, \quad \mathbf{b}_2 := \begin{bmatrix} (1 - \frac{h^2}{12} c_1)\alpha \\ 0 \\ \vdots \\ 0 \\ (1 - \frac{h^2}{12} c_{n+2})\beta \end{bmatrix}.$$

Save as `numerov3.m`. Test with $n = 3, \alpha = 1, \beta = -1$.

```
>> numerov3
b =
    1.4241
   -0.0018
   -1.4425
```

6. Compute and show \mathbf{v} , $\bar{\mathbf{v}}$ and $err = \|\bar{\mathbf{v}} - \mathbf{u}\|_\infty$. Also plot exact and approximate solutions. Save as `numerov4.m` and test with $n = 5$. The solution of (P) is $u(x) = \sin(\pi(x + 1/2))$. See Fig. 13.2.

```
>> numerov4
err =
  1.3792e-04
```

7. Create an example $f2$, $u2$, $c2$ such that $\|\bar{\mathbf{v}} - \mathbf{u}\|_\infty = 0$ up to the errors of the computer. Save as `numerov5.m`.
8. Back to the first example, write a program `numerovmorder.m` to study the error when modifying n . For the test, start with $n = [5, 10, 15, 20, 35, 50, 75, 100]$ and compute the corresponding $error$. Conclusion on the order. See Fig. 13.3.
9. Now we study the new problem

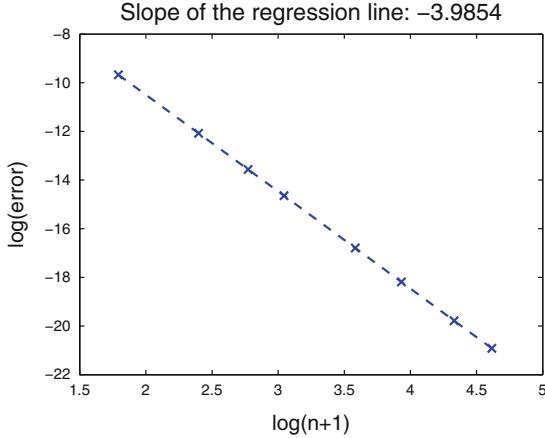


Fig. 13.3 Error between the exact solution and the Numerov approximation

$$(P') : \begin{cases} \frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} + c(x)u = 0, & x \in [0, 1], 0 \leq t \leq T \\ u(t, 0) = \alpha \\ u(t, T) = \beta \\ u(0, x) = \alpha + (\beta - \alpha)x \end{cases}$$

We approximate $\frac{\partial u}{\partial t}(t, x)$ by $\frac{u(t, x) - u(t - \tau, x)}{\tau}$ and use the notation in the Numerov scheme to obtain an implicit scheme

$$(A + \frac{h^2}{\tau} B)w^{k+1} = \frac{h^2}{\tau} B w^k + b_2, \text{ where } B = \frac{1}{12} \begin{bmatrix} 10 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & 1 & \\ 0 & & 1 & 10 \end{bmatrix} \in \mathbb{R}^{n \times n}$$

and w^k is an approximation of $[u((k-1)\tau, x_2) \dots u((k-1)\tau, x_{n+1})]^T$.

Write a program `numerov6.m` that computes an approximation of $u(T, x)$ for a given time step $\tau = T/k > 0$. Test with $n = 3, k = 5, T = 0.3, \alpha = 1, \beta = 2^{(1-\sqrt{5})/2}$ and $c_3(x) := \frac{1}{(x+1)^2}$.

10. The stationary solution which satisfies $\frac{\partial u}{\partial t} = 0$ is $u_\infty(x) := (x+1)^{\frac{1-\sqrt{5}}{2}}$. Show on a graph with several curves that $u(t, x)$ tends to $u_\infty(x)$. Test $n = 8, k = 5, T = 0.3$, discrete solutions at $0, \tau, \dots, T$ and stationary solution. Save as `numerov7.m` See Fig. 13.4.

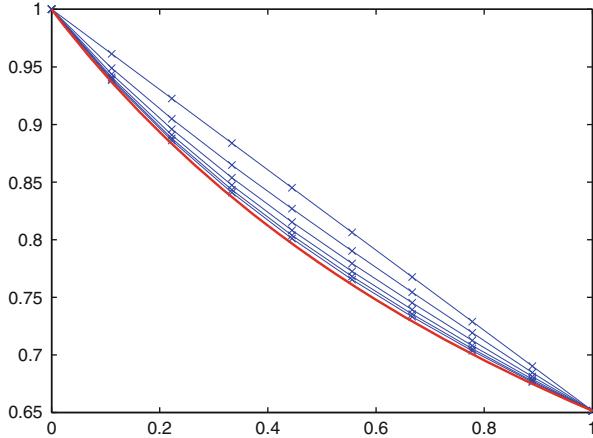


Fig. 13.4 Stationary solution and Numerov approximations at different times

Exercise 13.5. Convection-diffusion equation

In this exercise, we have chosen not to give details for the construction of the discrete problem and programs. Before working on this exercise, the reader should train with the previous exercises to become more autonomous in this one.

The goal of the exercise is the approximation of the solution of a convection-diffusion equation with conditions at the ends of the interval:

$$(P) \begin{cases} -u''(t) + rc(t)u'(t) = f(t), & t \in [0, 1] \\ u(0) = 0, u(1) = 0. \end{cases}$$

For a given positive integer n and $h := 1/(n+1)$, we define the uniform partition $t \in \mathbb{R}^{n+2}$ of $[0, 1]$ given by $t_i = (i-1)h$ for $i = 1, \dots, n+2$. For $i = 2, \dots, n+1$, we use the approximations of the derivatives deduced from (13.3) and (13.4).

1. Show that the discrete problem is given by the linear system

$$(P_h) : \left(\mathbf{A} + \frac{rh}{2} \mathbf{B} \right) \mathbf{v} = \mathbf{f}, \quad (13.12)$$

where

$$\mathbf{A} := \begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & 0 & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n \times n}, \quad \mathbf{f} := h^2 \begin{bmatrix} f(t_2) \\ \vdots \\ f(t_k) \\ \vdots \\ f(t_{n+1}) \end{bmatrix} \in \mathbb{R}^n \quad (13.13)$$

$$B := \begin{bmatrix} c(t_2) & 0 & 0 & \dots & 0 \\ 0 & c(t_3) & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & c(t_{n+1}) \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \dots & 0 & -1 & 0 \end{bmatrix} \in \mathbb{R}^{n \times n}. \quad (13.14)$$

The unknown $\mathbf{v} := [v_2, \dots, v_{n+1}]^T$ approximates $\mathbf{u} := [u(t_2), \dots, u(t_{n+1})]^T$ while $\bar{\mathbf{v}} := \begin{bmatrix} 0 \\ \mathbf{v} \\ 0 \end{bmatrix}$ is the approximate solution of (P) , at the points $\bar{\mathbf{u}}$ of the partition t .

In the following, for a given $r > 0$, we fix $f_r(t) = f1(r, t) := r^2 \frac{e^{r \cdot t} (t - 1)}{1 - e^r} + rt$ and $c1(t) := t$ so that the solution of (P) is $u_r(t) = u1(r, t) := t - \frac{1 - e^{r \cdot t}}{1 - e^r}$.

2. **Programming:** Write three function `f1.m` computing $f1(r, t)$, `u1.m` for $u1(r, t)$ and `c1.m` for $c1(t)$. Test

```
>> f1(1,0:3)
ans =
    0.5820    1.0000   -2.3003  -20.3786
```

```
>> u1(1,0:3)
ans =
    0         0     -1.7183   -8.1073
```

```
>> c1(0:3)
ans =
    0         1         2         3
```

3. Write a program `convdiff.m` that for a given n , computes the approximation $\bar{\mathbf{v}}$, plots it and also the exact solution $\bar{\mathbf{u}} := [u(r, t_1), \dots, u(r, t_{n+2})]^T$ then computes and shows $error = \|\bar{\mathbf{v}} - \bar{\mathbf{u}}\|_\infty$. Test with $r = 1$ and $n = 4$. Test also with $r = 100$ and $n = 10$. See Fig. 13.5

```
>> convdiff
r =
    1
n =
    4
bv =
    0
    0.0711
    0.1138
    0.1216
    0.0868
    0
error =
    7.8507e-05
```

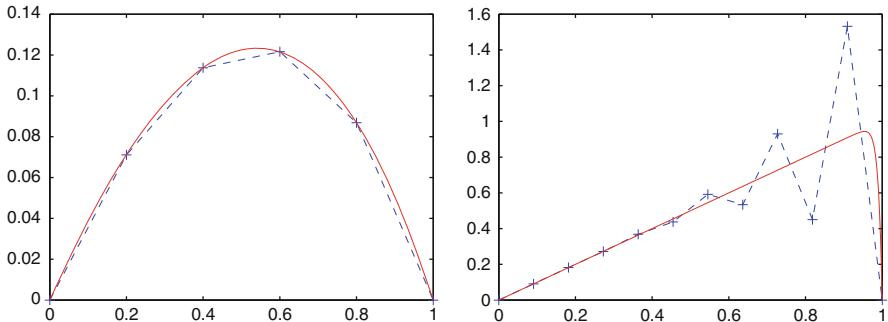


Fig. 13.5 Exact and discrete solutions with $r = 1$, $n = 4$ then $r = 100$, $n = 10$

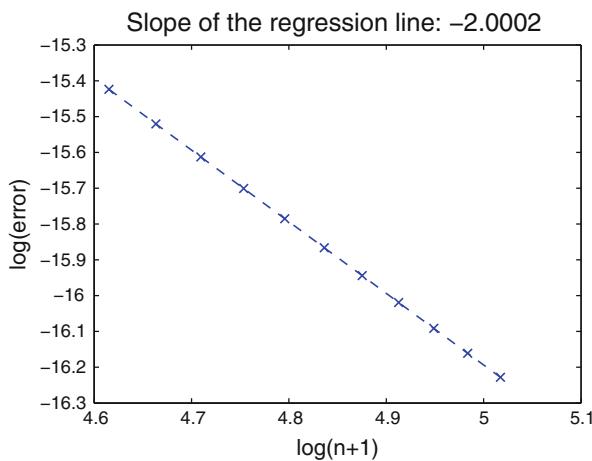


Fig. 13.6 Error with $r = 1$. The order is 2

4. We fix $r = 1$. Write a program to study the error when modifying n and estimate the order of the method. Save as `cdorder1.m`. See Fig. 13.6.
5. To avoid the error appearing for large values of r (see Fig. 13.5), we modify the discrete problem into:

$$(Q_h) : (\mathbf{A} + rh\mathbf{B}_1) \mathbf{v} = \mathbf{f} \quad (13.15)$$

where

$$\mathbf{B}_1 := \begin{bmatrix} c(t_2) & 0 & 0 & \dots & 0 \\ 0 & c(t_3) & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & c(t_{n+1}) \end{bmatrix}$$

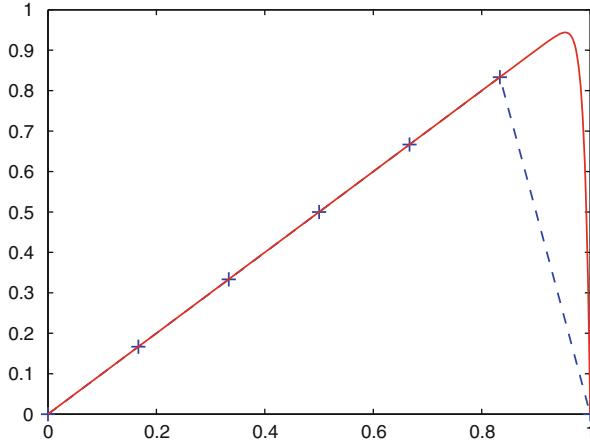


Fig. 13.7 Exact and new discrete solutions with $r = 100$ and $n = 5$

$$\times \begin{bmatrix} 2a_1 - 1 & 1 - a_1 & 0 & \dots & 0 \\ -a_2 & 2a_2 - 1 & 1 - a_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -a_{n-1} & 2a_{n-1} - 1 & 1 - a_{n-1} \\ 0 & \dots & 0 & -a_n & 2a_n - 1 \end{bmatrix}$$

$\mathbf{g} := rh[c(t_2), \dots, c(t_{n+1})]^T \in \mathbb{R}^n$ then $\mathbf{a} := 1/2 + 1/2 \coth(\mathbf{g}/2) - 1./\mathbf{g} \in \mathbb{R}^n$. Modify `convdiff.m` with the new matrix $\mathbf{M}_1 := \mathbf{A} + rh\mathbf{B}_1$. Save as `modifycd.m`. Test with $r = 100$ and $n = 10$. See Fig. 13.7. If $c(t_i) = 0$ then the corresponding a_i is not defined and we fix $a_i := 1/2$ in this case.

6. Study the error and the order starting from $arrn = [40 : 3 : 60]$ and $r = 100$. Save as `modifycdorder.m`. See Fig. 13.8.

13.3 Finite Differences in Dimension 2

Exercise 13.6. Laplacian on a square

We consider a physical problem satisfying a diffusion law. As an example, we consider a thin flexible plate subjected to the action of a vertical force f . We seek the vertical deflection u . More precisely, if f and g are two functions defined on $\Omega = [0, 1]^2$ with edge Γ and if g is defined on Γ , we look for a function u defined on Ω such that

$$(II) \quad \begin{cases} -\Delta u + cu = f \text{ on } \Omega \\ u|_{\Gamma} = g, \end{cases} \quad (13.16)$$

where $\Delta := \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian.

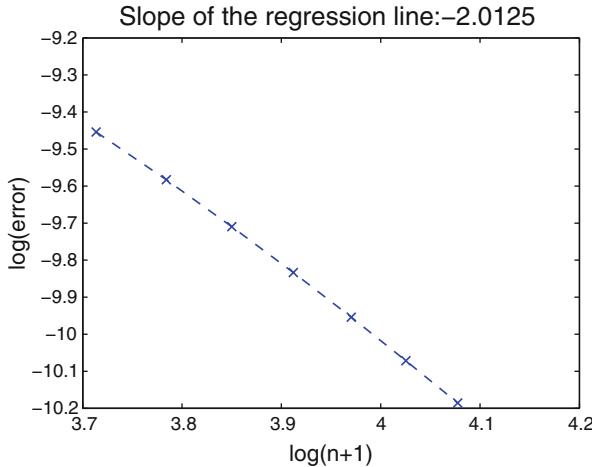


Fig. 13.8 Error for $r = 100$ with the new discrete problem

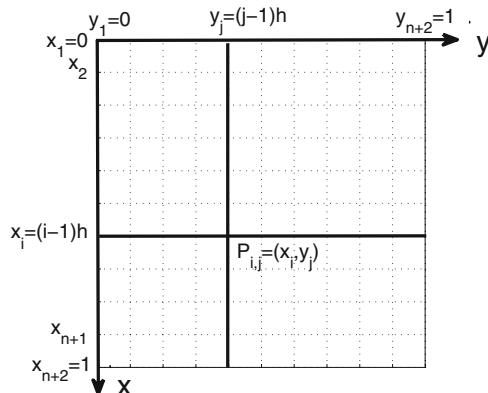


Fig. 13.9 The mesh with step $h = 1/(n + 1)$

Let n be a positive integer and $h := 1/(n + 1)$. We define a regular mesh $\Omega_h = \{(x_i, y_j) = ((i - 1)h, (j - 1)h)\}_{i,j=1}^{n+2}$ and let ω_h be the subset of all mesh points located in $(0, 1)^2$. It has cardinality n^2 . (see Fig. 13.9). We look for an approximation \bar{v} of the function u at the points of the mesh Ω_h such that

$$(II_h) \quad \begin{cases} -\Delta_h \bar{v}(P) + c(P) \bar{v}(P) = f(P) & \text{for } P \in \omega_h \\ \bar{v}(P) = g(P) & \text{for } P \in \Gamma, \end{cases} \quad (13.17)$$

where $\Delta_h \bar{v}$ is defined by (13.6)

1. Write the equations corresponding to $P(3h, 4h)$, $P(2h, 3h)$, $P(2h, 2h)$... for $n \geq 5$

2. Show that (Π_h) can be written

$$\mathbf{A}\mathbf{v} = \mathbf{f} + \mathbf{g} \quad (13.18)$$

where $\mathbf{A} \in \mathbb{R}^{n^2 \times n^2}$ with

$$\mathbf{A} := \begin{bmatrix} \mathbf{D}_2 & -\mathbf{I} & & \\ -\mathbf{I} & \mathbf{D}_3 & -\mathbf{I} & \\ & \ddots & \ddots & \ddots & O \\ & & \ddots & \ddots & -\mathbf{I} \\ O & & & -\mathbf{I} & \mathbf{D}_{n+1} \end{bmatrix}, \quad \mathbf{D}_j := \begin{bmatrix} d_{2,j} & -1 & & O \\ -1 & d_{3,j} & \ddots & \\ & \ddots & \ddots & -1 \\ O & & -1 & d_{n+1,j} \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$d_{ij} := 4 + h^2 c(x_i, y_j). \quad (13.19)$$

$\mathbf{f} \in \mathbb{R}^{n^2}$ with components $h^2 f(x_i, y_j)$, \mathbf{I} is the identity matrix of order n and $\mathbf{g} \in \mathbb{R}^{n^2}$ is the contribution from the points on Γ where $u = g$ is known.

3. Prove that \mathbf{A} is strictly diagonal dominant as soon as $c > 0$ and deduce that (Π_h) has a unique solution \triangleright Math Hint³ \triangleleft

⊕ **Comment:** If $u \in C^4(\Omega)$ then one can show (by an extension of the proof proposed in Exercise 13.3) that

$$\|u - u_h\|_\infty = \max_{P \in \Omega_h} |u(P) - U_h(P)| \leq c_1 h^2 M_4$$

where c_1 is a constant and $M_4 = \sup_{(x,y) \in \bar{\Omega}} \left(\left| \frac{\partial^4 \phi}{\partial x^4} \right|, \left| \frac{\partial^4 \phi}{\partial y^4} \right| \right)$. We will show numerically that this bound is optimal ⊕

Exercise 13.7. Computation on the square

We use the notation of Exercise 13.6. To solve the linear system (13.18) and to avoid the construction of the very large matrix \mathbf{A} with lots of zeros (a sparse matrix), we propose to use the Gauss-Seidel method where the computation only uses the nonzero elements. Let us recall that for $\mathbf{M}\mathbf{x} = \mathbf{b}$ with $\mathbf{M} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$, if we start by $\mathbf{x}^0 = \mathbf{0}$, then from $\mathbf{x}^p = [x_1^p, \dots, x_d^p]^T$, \mathbf{x}^{p+1} is defined by

$$x_k^{p+1} = \frac{1}{a_{kk}} \left(b_k - \sum_{j=1}^{k-1} m_{kj} x_j^{p+1} - \sum_{j=k+1}^d m_{kj} x_j^p \right) \text{ for } k = 1, \dots, d.$$

Instead of looking for a vector $\mathbf{v} \in \mathbb{R}^{n^2}$, we compute it in the form of a matrix $\mathbf{V} \in \mathbb{R}^{n \times n}$ approximating \mathbf{U} in such a way that $V_{i,j} \approx U_{i,j} = u(x_i, y_j)$ for $i, j = 2, \dots, n+1$. We add the values on the edges: $V_{i,j} = g(x_i, y_j)$ for $i, j = 1, n+1$. Thus $\bar{\mathbf{V}}$ is a matrix in $\mathbb{R}^{(n+2) \times (n+2)}$. Similarly, we will use the matrix $\mathbf{F}, \mathbf{C} \in \mathbb{R}^{n \times n}$ with $F_{i,j} = h^2 f(x_i, y_j)$ and $C_{i,j} = c(x_i, y_j)$ for $i, j = 2, \dots, n+1$. Finally $D_{i,j} = 4 + h^2 C_{i,j}$.

³ Use the results of Exercise 2.8.

1. After the initialization, show that step p of the Gauss-Seidel method can be written

```

for    i = 2 to n + 1
    for    j = 2 to n + 1
        
$$z = \frac{F_{ij} + V_{i-1,j} + V_{i,j-1} + V_{i+1,j} + V_{i,j+1}}{D_{i,j}}$$

        
$$err_{i,j} = |z - V_{i,j}|$$

        
$$V_{i,j} = z$$

    end j
end i

```

We iterate until convergence, defined here as $\max_{i,j} |err_{i,j}| < precis$, where $precis$ is a given precision, or until the maximum number of iterations is reached defined here as divergence.

For a given positive integer n

- Generate the matrices $[D_{ij}]$ and $[F_{ij}]$ using .m functions,
- Initialize \bar{V} with

$$\begin{aligned} V_{i,j} &= 0 \text{ for } i, j = 2, \dots, n + 1, \\ V_{i,j} &= g(x_i, y_j) \text{ for } i, j = 1, n + 2 \end{aligned}$$

- Give values for $maxiter$ and $precis$,

- Apply the Gauss-Seidel method,

- If the method converges, the approximation is in \bar{V} .

Program the algorithm for a given integer n . Validate the program with an example where $\Delta_h u = \Delta u$ so that the solutions of the exact and discrete problems, \bar{U} and \bar{V} coincides. The first step is the construction of the mesh $\{(x_i, y_j)\}$. ◁ MATLAB Hint⁴ ▷ The end of the loop (Gauss-Seidel iterations) depends on two parameters, use $maxiter = 30 \times n$ and $precis = 10^{-10}$. ◁ MATLAB Hint⁵ ▷ Save as approxlaplac1.m

- Now we choose $c_2(x, y) := 2\pi^2$, $f_2(x, y) := 7\pi^2 \sin \pi(2x + y)$ so that the solution of (Π) is $u_2(x, y) = \sin \pi(2x + y)$ which also gives g on the edges. Add the graphs of the exact and discrete solutions. Save as approxlaplac2.m. Test with $n = 5$. See Fig. 13.10.

```

>> approxlaplac2
n =
    10
maxiter =
    300
iter =
    126
error =
    0.0186

```

⁴ Use `meshgrid`.

⁵ Use `while`.

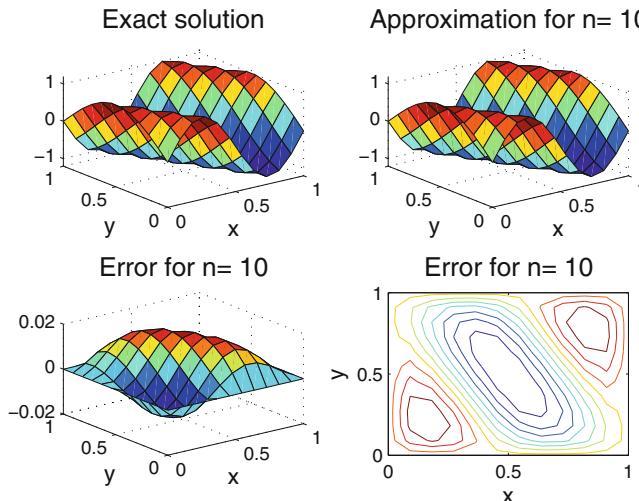


Fig. 13.10 Exact and discrete solutions with $r = 100$

4. Study $\|\mathbf{U} - \mathbf{V}\|_\infty \times (n + 1)^2$ when modifying n . Start with the array $\mathbf{n} = 60 : 65$, $\text{maxiter} = 50 \times n$ and $\text{precis} = 10^{-8}$. Conclusion on the order. Save as `orderlaplac.m`.

```
>> orderlaplace
error*(n+1)^2
ans =
    2.2522      2.2525      2.2513      2.2515      2.2502      2.2503
```

Exercise 13.8. A nonlinear problem

Consider the weakly nonlinear problem

$$(II') \quad \begin{cases} -\Delta u(x, y) + \sin(u(x, y)) = f(x, y), & (x, y) \in \Omega \\ u |_{\Gamma} = 0 \end{cases}$$

We keep the notations of Exercise 13.7 and, again, we use an iterative method

$$4V_{i,j} = h^2 f(x_i, y_j) + V_{i-1,j} + V_{i+1,j} + V_{i,j-1} + V_{i,j+1} - h^2 \sin(V_{i,j}).$$

1. Write an iterative algorithm.
2. The function f is defined by $f3(x, y) := 5\pi^2 \sin(\pi x) \sin(2\pi y) + \sin(\sin(\pi x) \sin(2\pi y))$, so that the solution of (II') is $u3(x, y) := \sin(\pi x) \sin(2\pi y)$. Create the functions `f3.m` and `u3.m`, then modify `approxlaplac2.m` to adapt the computation to the new algorithm. Save as `approxlaplacnonlin.m`. Test with $n = 10$.
3. Study the error when modifying n and find the order of the method. Save as `orderlaplacenonlin.m`. Test with $\text{tabn} = 60 : 65$.

13.4 The Heat Equation and Approximations

Exercise 13.9. Heat equation

Consider a bar of length L in which heat is transmitted by conduction. It is assumed that the bar also has heat exchange with the outside. Precisely, the amount of heat supplied at point x at time t is $f(x, t)$. The temperature $\theta(x, t)$ then satisfies

$$\sigma \frac{\partial \theta}{\partial t}(x, t) - \gamma \frac{\partial^2 \theta}{\partial x^2}(x, t) = f(x, t)$$

where σ is the specific heat and γ is the heat conductivity. Assume we know the initial temperature $\theta(x, 0)$ and also temperatures at the ends $\theta(0, t) = \alpha(t)$ and $\theta(L, t) = \beta(t)$.

By a change of variables, we can assume $\sigma = \gamma = L = 1$ and $\alpha(t) = \beta(t) = 0$, so that the problem is now to find u such that

$$(P) \quad \begin{cases} \frac{\partial u}{\partial t}(x, t) - \frac{\partial^2 u}{\partial x^2}(x, t) = f(x, t), & 0 \leq x \leq 1, 0 \leq t \\ u(x, 0) = u_0(x), & 0 \leq x \leq 1, \quad \text{initial conditions} \\ u(0, t) = u(1, t) = 0, & 0 \leq t, \quad \text{end conditions.} \end{cases} \quad (13.20)$$

For compatibility reasons, we assume that $u_0(0) = u_0(1) = 0$ and also that f and u_0 are smooth enough functions. In this case, (P) has a unique solution that will be written u .

Approximate problem

We look for a discrete solution v defined on a mesh in space and time. For a positive integer n , we set the space step at $\Delta x = h := \frac{1}{n+1}$ and the time step at $k = \Delta t$. We wish to compute v at $(x_i, t_\ell) := ((i-1)h, (\ell-1)\Delta t)$ for $i = 1, \dots, n+2$ and $1 \leq \ell$. If we set $V(i, n) = v(x_i, t_n)$ and use the known values so that $V(i, 1) = u_0(x_i)$, $V(1, \ell) = V(n+2, \ell) = 0$.

We write $\bar{V}(:, \ell) = [v(x_i, t_\ell)]_{i=1}^{n+2}$ and $V(:, \ell) = [v(x_i, t_\ell)]_{i=2}^{n+1}$. We want to compute $V(:, \ell)$ for $\ell \geq 2$.

For a smooth enough function ϕ defined on a subset of \mathbb{R}^2 , using (13.1), (13.2) and (13.4), we approximate the partial derivatives by:

$$\frac{\partial \phi}{\partial t}(x, t) \approx \frac{\phi(x, t + \Delta t) - \phi(x, t)}{\Delta t} \quad (13.21)$$

$$\frac{\partial \phi}{\partial t}(x, t) \approx \frac{\phi(x, t) - \phi(x, t - \Delta t)}{\Delta t} \quad (13.22)$$

$$\frac{\partial^2 \phi}{\partial x^2}(x, t) \approx \frac{\phi(x - h, t) - 2\phi(x, t) + \phi(x + h, t)}{h^2} \quad (13.23)$$

1. With these approximations of the partial derivatives, if we know $V(:, \ell)$, we will compute $V(:, \ell + 1)$ by the following three methods:

a. **Explicit Scheme.** We use formulas (13.21) and (13.23) for u at the point (x_i, t_ℓ) . Find $V(i, \ell + 1)$ depending of the $V(j, \ell)$ $j = i - 1, i, i + 1$, then show that

$$V(:, \ell + 1) = \left(\mathbf{I} - \frac{\Delta t}{h^2} \mathbf{A} \right) V(:, \ell) + \Delta t F(:, \ell), \quad (13.24)$$

where $\mathbf{A} := \begin{bmatrix} 2 & -1 & & O \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ O & & -1 & 2 \end{bmatrix} \in \mathbb{R}^{n \times n}$ and \mathbf{I} is the identity matrix.

b. **Implicit Scheme.** We use (13.22) and (13.23) for u at the point $(x_i, t_{\ell+1})$. Find an equation linking $V(i - 1, \ell + 1), V(i, \ell + 1), V(i + 1, \ell + 1)$ and $V(i, \ell)$ and deduce that

$$\left(\mathbf{I} + \frac{\Delta t}{h^2} \mathbf{A} \right) V(:, \ell + 1) = V(:, \ell) + \Delta t F(:, \ell + 1). \quad (13.25)$$

Prove that the system has a unique solution so that $V(:, \ell + 1)$ can be computed from $V(:, \ell)$. ▷ *Math Hint*⁶ ◁

c. **Crank-Nicolson scheme.**

John Crank (1916–2006) was an English mathematical physicist. He worked on ballistics during the Second World War and its main work was on the numerical solution of partial differential equations and, in particular, the solution of heat-conduction problems.



Phyllis Nicolson (1917–1968) was a British mathematician graduated from Manchester University and then was a research fellow at Girton College, Cambridge. She is most known for her work on the Crank-Nicolson scheme together with John Crank.



We compute an average of the formulas (13.24) and (13.25):

$$\left(\mathbf{I} + \frac{\Delta t}{2h^2} \mathbf{A} \right) V(:, \ell + 1) = \left(\mathbf{I} - \frac{\Delta t}{2h^2} \mathbf{A} \right) V(:, \ell) + \frac{\Delta t}{2} (F(:, \ell) + F(:, \ell + 1)). \quad (13.26)$$

⁶ Use Theorem 2.1 that gives properties of $(\mathbf{I} + \frac{\Delta t}{h^2} \mathbf{A})$.

2. The order of the methods: We define $U(i, \ell) = u(x_i, t_\ell)$, where u is the exact solution. For the explicit method, from (13.24), we deduce that

$$V(:, \ell + 1) - V(:, \ell) + \frac{\Delta t}{h^2} A V(:, \ell) - \Delta t F(:, \ell) = 0,$$

and we define the consistency error

$$\epsilon_1(x, t) := u(x, t + \Delta t) - u(x, t) - \frac{\Delta t}{h^2} [u(x - h, t) - 2u(x, t) + u(x + h, t)] - \Delta t f(x, t).$$

Similarly from (13.25) then (13.26), we define

$$\begin{aligned} \epsilon_2(x, t) &:= u(x, t + \Delta t) - u(x, t) \\ &\quad - \frac{\Delta t}{h^2} [u(x - h, t + \Delta t) - 2u(x, t + \Delta t) + u(x + h, t + \Delta t)] \\ &\quad - \Delta t f(x, t + \Delta t) \end{aligned}$$

$$\begin{aligned} \epsilon_3(x, t) &:= u(x, t + \Delta t) - u(x, t) - \frac{\Delta t}{2h^2} [u(x - h, t) - 2u(x, t) + u(x + h, t) \\ &\quad + u(x - h, t + \Delta t) - 2u(x, t + \Delta t) + u(x + h, t + \Delta t)] \\ &\quad - \frac{\Delta t}{2} [f(x, t) + f(x, t + \Delta t)]. \end{aligned}$$

Find a bound of $|\epsilon_j(x, t)|$ for $j = 1, 2, 3$ as a function of Δt , h and the maxima of the partial derivatives of u . ▷ *Math Hint*⁷ ◁

3. Compare the bounds in the case where $h = 0(\Delta t)$ and deduce the orders of the methods.
4. **Example:** We now set $f := 0$. The odd extension of the function u_0 can be expanded in a Fourier sine series involving $\sin p\pi x$, $p = 1, 2, 3, \dots$ (cf. Exercise 11.7). Without loss of generality, let us suppose that $u_0(x) := \sin p\pi x$ for some $p \in \mathbb{N}$.
 - a. Show directly (without using (11.14)) that the solution of (P) is $u(x, t) = \sin(p\pi x) e^{-p^2\pi^2 t}$.
 - b. We recall that

$$\sin[(i-1)\theta] - 2\sin(i\theta) + \sin[(i+1)\theta] = -4\sin^2 \frac{\theta}{2} \sin(i\theta).$$

Deduce that for the three schemes $V(i, n) = \alpha^n \sin(p\pi i h)$ where α is a constant depending of the scheme, h and Δt .

- c. Find conditions on h and Δt such that the discrete solutions is bounded when ℓ tends to infinity.

© **Comment:** In this case, the scheme is **stable** and the method is convergent. ◉

⁷ Use (13.1), (13.2) and (13.4) for u at (x, t) and $(x, t + \Delta t)$.

Exercise 13.10. Heat programs

We start from the simplified heat equation:

$$(P) \begin{cases} \frac{\partial u}{\partial t}(x, t) - \frac{\partial^2 u}{\partial x^2}(x, t) = 0, & 0 \leq x \leq 1, 0 \leq t \leq T, \\ u(x, 0) = u_0(x), & 0 \leq x \leq 1, \\ u(0, t) = u(1, t) = 0, & 0 \leq t. \end{cases}$$

We assume that $u_0(x) := \sin(\pi x)$, thus the solution is $u(x, t) = \sin(\pi x)e^{-\pi^2 t}$. In the following, we assume that $T = 0.4$. For the approximation, we use the **Implicit method** with the notations of the previous exercise.

1. Write a program `heq1.m` that given n , computes $V(:, 1)$, and the matrix A . Test with $n = 3$.

```
>> heq1
V =
    0.7071
    1.0000
    0.7071
A =
    2     -1      0
   -1      2     -1
     0     -1      2
```

2. Extend the first program to `heq2.m` with the computation of $V(:, \ell)$ and \bar{V} then the graphs of the exact and discrete solutions. Also plot the level curves.
 ◁ MATLAB Hint⁸ ▷ Test with $n = 6$ and $\Delta t = 0.1$ then $n = 10$ and $\Delta t = 0.02$. See Fig. 13.11.

```
>> heq2
bV =
    0         0         0         0         0
    0.4339   0.2202   0.1117   0.0567   0.0288
    0.7818   0.3968   0.2014   0.1022   0.0519
    0.9749   0.4948   0.2511   0.1274   0.0647
    0.9749   0.4948   0.2511   0.1274   0.0647
    0.7818   0.3968   0.2014   0.1022   0.0519
    0.4339   0.2202   0.1117   0.0567   0.0288
    0         0         0         0         0
```

3. We fix $n = 200$. Study the error between the exact and discrete solution when modifying Δt . Begin with $arrdt = [0.01, 0.005, 0.002, 0.001]$ and plot

⁸ Use `meshgrid`, `surf`, `contour`.

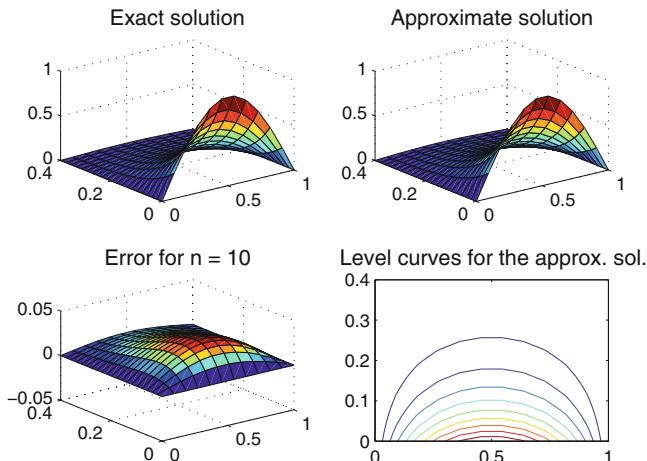


Fig. 13.11 Exact and discrete solutions in the heat equation $n = 10$, $\Delta t = 0.02$

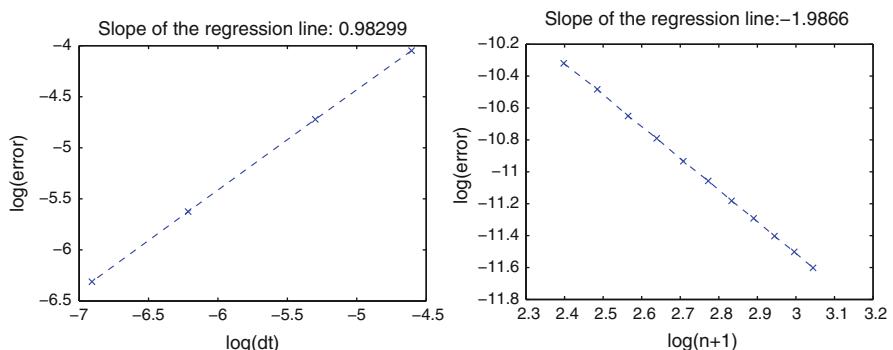


Fig. 13.12 Error for the heat equation: order 1 for time and 2 for space

$\log(\text{error})$ function of $\log(\Delta t)$. Save as `heq3.m`. Conclusion on the order in t . See Fig. 13.12.

4. We fix $\Delta t = 0.000001$ and $T = 0.0005$. Study the error between the exact and discrete solution when modifying $h = \Delta x$. Begin from $arrn = 10 : 20$ and plot $\log(\text{error})$ function of $\log(n + 1)$. Save as `heq4.m`. Conclusion on the order in x . See Fig. 13.12.
5. Restart from the beginning with the **Crank-Nicolson scheme**. See Fig. 13.13 for the error when Δt is modified. Use the value $n = 500$ and $arrdt = [0.1, 0.05, 0.02, 0.01, 0.005, 0.002]$.

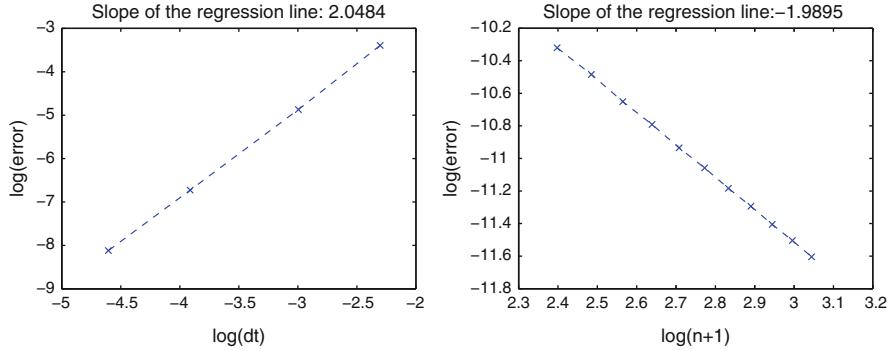


Fig. 13.13 Error for the Crank-Nicolson scheme: order 2 for time and 2 for space

13.5 Solutions

13.5.1 Definitions of Finite Differences

Solutions of Exercise 13.1

By first order Taylor expansion, there exists $\theta_1 \in (0, 1)$ such that $f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x+\theta_1 h)$ from which we deduce $\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{h}{2}f''(x+\theta_1 h)$ and (13.1) follows. (13.2) is obtained similarly.

For (13.3) we use a Taylor expansion of order 3 with h and the same one with $-h$:

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x+\alpha_1 h), \quad \alpha_1 \in (0, 1)$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x-\alpha_2 h), \quad \alpha_2 \in (0, 1).$$

Then we compute the difference of the two previous lines:

$$f(x+h) - f(x-h) = 2hf'(x) + \frac{h^3}{3} \frac{f^{(3)}(x+\alpha_1 h) + f^{(3)}(x-\alpha_2 h)}{2}.$$

By the Intermediate value Theorem $\frac{f^{(3)}(x+\alpha_1 h) + f^{(3)}(x-\alpha_2 h)}{2} = f^{(3)}(x+\theta_2 h)$ with $\theta_2 \in (-1, 1)$ and we deduce (13.3).

For the last formula, there exist $\beta_1, \beta_2 \in (0, 1)$ such that

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x+\beta_1 h),$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f^{(3)}(x) + \frac{h^4}{24}f^{(4)}(x-\beta_2 h).$$

We compute the sum and (13.4) is obtained by the Intermediate value Theorem again.

Solutions of Exercise 13.2

1. Taylor expansions of order 3 for $i = 1$ and 3 give the existence of $h'_i \in (0, h_i)$ such that

$$\begin{aligned}\phi(x + h_1, y) &= \phi(x, y) + h_1 \frac{\partial \phi}{\partial x}(x, y) + \frac{h_1^2}{2} \frac{\partial^2 \phi}{\partial x^2}(x, y) + \frac{h_1^3}{6} \frac{\partial^3 \phi}{\partial x^3}(x + h'_1, y), \\ \phi(x - h_3, y) &= \phi(x, y) - h_3 \frac{\partial \phi}{\partial x}(x, y) + \frac{h_3^2}{2} \frac{\partial^2 \phi}{\partial x^2}(x, y) - \frac{h_3^3}{6} \frac{\partial^3 \phi}{\partial x^3}(x - h'_3, y),\end{aligned}$$

from which we deduce that

$$\begin{aligned}&\frac{\phi(P_1) - \phi(P)}{h_1} + \frac{\phi(P_3) - \phi(P)}{h_3} \\ &= \frac{h_1 + h_3}{2} \frac{\partial^2 \phi}{\partial x^2}(P) + \frac{h_1^2}{6} \frac{\partial^3 \phi}{\partial x^3}(x + h'_1, y) - \frac{h_3^2}{6} \frac{\partial^3 \phi}{\partial x^3}(x - h'_3, y).\end{aligned}$$

We divide by $(h_1 + h_3)/2$ and find

$$\begin{aligned}\frac{\partial^2 \phi}{\partial x^2}(P) &= 2 \frac{\phi(P_1) - \phi(P)}{h_1(h_1 + h_3)} + 2 \frac{\phi(P_3) - \phi(P)}{h_3(h_1 + h_3)} \\ &\quad - \frac{h_1^2}{3(h_1 + h_3)} \frac{\partial^3 \phi}{\partial x^3}(x + h'_1, y) + \frac{h_3^2}{3(h_1 + h_3)} \frac{\partial^3 \phi}{\partial x^3}(x - h'_3, y),\end{aligned}$$

from which we obtain

$$\begin{aligned}&\left| \frac{\partial^2 \phi}{\partial x^2}(P) - \frac{2}{h_1(h_1 + h_3)} \phi(P_1) - \frac{2}{h_3(h_1 + h_3)} \phi(P_3) + \frac{2}{h_1 h_3} \phi(P) \right| \\ &\leq \frac{h_1^2 + h_3^2}{3(h_1 + h_3)} M_3 \leq \frac{h}{3} M_3 \text{ where } h = \max h_i.\end{aligned}$$

The last inequality follows if $\mu := h_3/h_1 \leq 1$ since then

$$\frac{h_1^2 + h_3^2}{h_1 + h_3} = h_1 \frac{1 + \mu^2}{1 + \mu} \leq h_1 \leq h$$

and for $h_1 \leq h_3$ by symmetry. A similar bound in the variable y can be computed. Then

$$|\Delta \phi(P) - \Delta_h \phi(P)| \leq c_1 h M_3, \quad c_1 = \frac{2}{3}.$$

Comment: In fact, the calculations are performed for functions of one variable since we work in one direction at a time. ☺

2. If $h = h_1 = h_2 = h_3 = h_4$, using a Taylor expansion of order 4 or directly (13.4), we obtain that

$$\frac{\partial^2 \phi}{\partial x^2}(P) = \frac{\phi(P_1) + \phi(P_3) - 2\phi(P)}{h^2} - \frac{h^2}{12} \frac{\partial^4 \phi}{\partial x^4}(P''_1),$$

$$\frac{\partial^2 \phi}{\partial y^2}(P) = \frac{\phi(P_2) + \phi(P_4) - 2\phi(P)}{h^2} - \frac{h^2}{12} \frac{\partial^4 \phi}{\partial y^4}(P''_2),$$

and $|\Delta\phi(P) - \Delta_h\phi(P)| \leq \frac{h^2}{6} M_4$, where $M_4 := \max_{(x,y) \in R} \left(\left| \frac{\partial^4 \phi}{\partial x^4} \right|, \left| \frac{\partial^4 \phi}{\partial y^4} \right| \right)$.

13.5.2 Applications of Finite Differences in Dimension 1

Solutions of Exercise 13.3

1. The differential equation $-u'' + cu = f$ is linear with constant coefficients. Since $c > 0$, it can be written $c = \omega^2$ with $\omega > 0$. Thus, the set of solutions of the homogeneous equation $-u'' + cu = 0$ is $u(x) = \lambda e^{\omega x} + \mu e^{-\omega x}$, with $\lambda, \mu \in \mathbb{R}$. Using variation of the constant we look for a particular solution of $-u'' + cu = f$ in the form $u(x) = \lambda(x)e^{\omega x}$. The function λ satisfies

$$\lambda''(x) + 2\omega\lambda'(x) = -e^{-\omega x}f(x) \quad (13.27)$$

which is a linear equation of order one in λ' . The solutions of $(\lambda')'(x) + 2\omega\lambda'(x) = 0$ are of the form $\lambda'(x) = \nu e^{-2\omega x}$ with $\nu \in \mathbb{R}$. Now $\lambda'(x) := e^{-2\omega x}\nu(x)$ satisfies (13.27) provided $\nu'(x) = -e^{\omega x}f(x)$, and since f is continuous this holds if $\nu(x) = -\int_0^x e^{\omega u}f(u)du$.

It follows that $\lambda(x) := -\int_0^x \left(e^{-2\omega t} \int_0^t e^{\omega u}f(u)du \right) dt$ is a solution of (13.27) and finally the solutions of $-u'' + cu = f$ are given by

$$u(x) = \lambda e^{\omega x} + \mu e^{-\omega x} - e^{\omega x} \int_0^x \left(e^{-2\omega t} \int_0^t e^{\omega u}f(u)du \right) dt.$$

If we fix $u(0) = u(1) = 0$, then we obtain a linear system with two unknowns λ and μ :

$$\begin{cases} \lambda + \mu = 0 \\ \lambda e^\omega + \mu e^{-\omega} = e^\omega \int_0^1 \left(e^{-2\omega t} \int_0^t e^{\omega u}f(u)du \right) dt \end{cases}$$

Since its determinant is nonzero it has a unique solution from which we deduce that (P) also has a unique solution.

2. For $i = 2, \dots, n+1$, in (13.7), we replace $u''(x_i)$ by

$$\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} - \frac{h^2}{12}u^{(4)}(x_i + \theta_4 h). \text{ Then}$$

$$-\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} + \frac{h^2}{12}u^{(4)}(x_i + \theta_4 h) + cu(x_i) = f(x_i).$$

If we ignore the 4th derivative term and denote the resulting approximations by $v_j \approx u(x_j)$ for $j = i-1, i, i+1$, the corresponding equation is

$$-\frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + cv_i = f(x_i), \quad i = 2, \dots, n+1.$$

3. We multiply this last equation by h^2 . Since $v_1 = 0$ and $v_{n+2} = 0$ we obtain the system

$$\left\{ \begin{array}{lllll} & +(2 + ch^2)v_2 & -v_3 & = & h^2f(x_2) \\ -v_2 & +(2 + ch^2)v_3 & -v_4 & = & h^2f(x_3) \\ \vdots & \vdots & \vdots & & \vdots \\ -v_{i-1} & +(2 + ch^2)v_i & -v_{i+1} & = & h^2f(x_i) \\ \vdots & \vdots & \vdots & & \vdots \\ -v_{n-1} & +(2 + ch^2)v_n & -v_{n+1} & = & h^2f(x_n) \\ -v_n & +(2 + ch^2)v_{n+1} & & & = h^2f(x_{n+1}) \end{array} \right.$$

and (13.8) follows.

4. a. Let $\mathbf{A}_h \mathbf{v} \geq \mathbf{0}$. We fix i such that $v_i := \min_{j=2, \dots, n+1} v_j$. First, we suppose that $i \in [3, n]$. By assumption $-v_{i-1} + (2 + h^2c)v_i - v_{i+1} \geq 0$ so that $v_i - v_{i-1} + h^2cv_i + v_i - v_{i+1} \geq 0$. Since $v_i - v_{i-1} \leq 0$ and $v_i - v_{i+1} \leq 0$ we deduce that $h^2cv_i \geq 0$. Now $h^2c > 0$ by the hypothesis on c and it follows that $v_i \geq 0$. We have a similar argument for $i = 2$ and $i = n+1$.

In every case, $v_i = \min_{j=2, \dots, n+1} v_j \geq 0$ and hence $\mathbf{v} \geq \mathbf{0}$.

- b. If $\mathbf{A}_h \mathbf{v} = \mathbf{0}$ then $\mathbf{A}_h \mathbf{v} \geq \mathbf{0}$, thus $\mathbf{v} \geq \mathbf{0}$. But also $-\mathbf{A}_h \mathbf{v} = \mathbf{A}_h(-\mathbf{v}) = \mathbf{0}$ then $\mathbf{A}_h(-\mathbf{v}) \geq \mathbf{0}$ thus $-\mathbf{v} \geq \mathbf{0}$. By both inequalities, we deduce that $\mathbf{v} = \mathbf{0}$. Hence $\mathbf{A}_h \mathbf{v} = \mathbf{0} \Rightarrow \mathbf{v} = \mathbf{0}$ which tells us that \mathbf{A}_h is nonsingular. We deduce that (P_h) has a unique solution.

Since $\mathbf{A}_h \mathbf{A}_h^{-1} = \mathbf{I}$ we obtain $\mathbf{A}_h \mathbf{c}_j = \mathbf{e}_j$, where $\mathbf{A}_h^{-1} = [\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_n]$ and \mathbf{e}_j is the j -th vector of the natural basis. Since $\mathbf{e}_j \geq \mathbf{0}$, we deduce that $\mathbf{c}_j \geq \mathbf{0}$. Thus $\mathbf{A}_h^{-1} \geq \mathbf{0}$.

5. Error:

- a. $\psi(x) = \frac{x(1-x)}{2h^2} \|\boldsymbol{\theta}\|_\infty$ is a polynomial that satisfies $\psi(x_1) = \psi(0) = 0$, $\psi(x_{n+2}) = \psi(1) = 0$ and has second derivative $\psi''(x) = -\frac{\|\boldsymbol{\theta}\|_\infty}{h^2}$. Also $\psi^{(4)} = 0$ from which we deduce from (13.4) that for $j = 2, \dots, n+1$,

$$\|\boldsymbol{\theta}\|_\infty = -h^2\psi''(x_j) = -\psi(x_{j-1}) + 2\psi(x_j) - \psi(x_{j+1}).$$

Hence

$$\mathbf{A}_h \Psi = \begin{bmatrix} \|\theta\|_\infty + h^2 c \psi(x_2) \\ \vdots \\ \|\theta\|_\infty + h^2 c \psi(x_i) \\ \vdots \\ \|\theta\|_\infty + h^2 c \psi(x_{n+1}) \end{bmatrix} \geq \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_i \\ \vdots \\ \theta_n \end{bmatrix} = \mathbf{A}_h w.$$

Also $\|\theta\|_\infty + h^2 c \psi(x_{i+1}) \geq -\theta_i$ and $\mathbf{A}_h \Psi \geq \mathbf{A}_h(-w)$. Then we have $\mathbf{A}_h(\Psi - w) \geq \mathbf{0}$ thus $\Psi - w \geq \mathbf{0}$ and $\mathbf{A}_h(\Psi + w) \geq \mathbf{0}$ thus $\Psi + w \geq \mathbf{0}$ by the properties of \mathbf{A}_h .

- b. The previous results give $w_i \leq \psi_i$ and $-w_i \leq \psi_i$ for all $i = 1, \dots, n$, so that $|w_i| \leq |\psi_i| \leq \|\Psi\|_\infty$ and finally $\|w\|_\infty \leq \|\Psi\|_\infty \leq \frac{1}{8h^2} \|\theta\|_\infty$.
- c. From (13.4), for $i = 2, \dots, n+1$,

$$\begin{aligned} -u(x_{i-1}) &+ (2 + ch^2)u(x_i) - u(x_{i+1}) \\ &= -h^2 u''(x_i) + h^2 cu(x_i) + \frac{h^4}{12} u^{(4)}(x'_i) \\ &= h^2 f(x_i) + \frac{h^4}{12} u^{(4)}(x'_i). \end{aligned}$$

Since $-v_{i-1} + (2 + ch^2)v_i - v_{i+1} = h^2 f(x_i)$, we deduce that

$$\mathbf{A}_h(w_h - v_h) = \frac{h^4}{12} [u^{(4)}(x'_2), \dots, u^{(4)}(x'_{n+1})]^T.$$

- d. Let $w := u_h - v_h$ be the error and let us define $\theta := \frac{h^4}{12} [u^{(4)}(x'_2), \dots, u^{(4)}(x'_{n+1})]^T$. Then $\mathbf{A}_h w = \theta$ and from the previous questions, we obtain that

$$\|u_h - v_h\|_\infty = \|w\|_\infty \leq \frac{1}{8h^2} \|\theta\|_\infty \leq \frac{h^2}{96} M_4 \text{ where } M_4 = \max_{x \in [0,1]} |u^{(4)}(x)|.$$

Comment: The error on the approximation of u'' was of order h^2 and we keep the same order even after the solution of the system. A numerical computation will show that the condition number (cf. Chap. 4) of \mathbf{A}_h is not increasing with n . ☀

Solutions of Exercise 13.4

1. Again, we use Taylor's expansions, there exist $\alpha_i \in (0, 1)$, $i = 1, 2$ such that:

$$\begin{aligned} \varphi(x+h) &= \varphi(x) + h\varphi'(x) + \frac{h^2}{2}\varphi''(x) + \frac{h^3}{3!}\varphi^{(3)}(x) + \frac{h^4}{4!}\varphi^{(4)}(x) \\ &\quad + \frac{h^5}{5!}\varphi^{(5)}(x) + \frac{h^6}{6!}\varphi^{(6)}(x + \alpha_1 h) \end{aligned}$$

$$\begin{aligned}\varphi(x-h) &= \varphi(x) - h\varphi'(x) + \frac{h^2}{2}\varphi''(x) - \frac{h^3}{3!}\varphi^{(3)}(x) + \frac{h^4}{4!}\varphi^{(4)}(x) \\ &\quad - \frac{h^5}{5!}\varphi^{(5)}(x) + \frac{h^6}{6!}\varphi^{(6)}(x - \alpha_2 h)\end{aligned}$$

We deduce that

$$\begin{aligned}\frac{\varphi(x-h) - 2\varphi(x) + \varphi(x+h)}{h^2} \\ = \varphi''(x) + \frac{h^2}{12}\varphi^{(4)}(x) + \frac{h^4}{6!}(\varphi^{(6)}(x + \alpha_1 h) + \varphi^{(6)}(x - \alpha_2 h)).\end{aligned}\tag{13.28}$$

Similarly, there exist $\alpha_i \in (0, 1)$, $i = 3, 4$ such that:

$$\begin{aligned}\varphi''(x+h) &= \varphi''(x) + h\varphi^{(3)}(x) + \frac{h^2}{2!}\varphi^{(4)}(x) + \frac{h^3}{3!}\varphi^{(5)}(x) + \frac{h^4}{4!}\varphi^{(6)}(x + \alpha_3 h) \\ \varphi''(x-h) &= \varphi''(x) - h\varphi^{(3)}(x) + \frac{h^2}{2!}\varphi^{(4)}(x) - \frac{h^3}{3!}\varphi^{(5)}(x) + \frac{h^4}{4!}\varphi^{(6)}(x - \alpha_4 h)\end{aligned}$$

so that

$$\begin{aligned}\varphi''(x-h) + 10\varphi''(x) + \varphi''(x+h) \\ = 12\varphi''(x) + h^2\varphi^{(4)}(x) + \frac{h^4}{4!}(\varphi^{(6)}(x + \alpha_3 h) + \varphi^{(6)}(x - \alpha_4 h)).\end{aligned}\tag{13.29}$$

Computing the difference (13.28)–(13.29), we obtain

$$\begin{aligned}&\left| \frac{\varphi(x-h) - 2\varphi(x) + \varphi(x+h)}{h^2} - \frac{\varphi''(x-h) + 10\varphi''(x) + \varphi''(x+h)}{12} \right| \\ &\leq \frac{h^4}{6!} |\varphi^{(6)}(x + \alpha_1 h) + \varphi^{(6)}(x - \alpha_2 h)| \\ &\quad + \frac{h^4}{12 \times 4!} |\varphi^{(6)}(x + \alpha_3 h) + \varphi^{(6)}(x - \alpha_4 h)| \\ &\leq \gamma h^4 \sup_{t \in [x-h, x+h]} |\varphi^{(6)}(t)| \text{ with } \gamma = \frac{7}{720}.\end{aligned}$$

2. Replacing $\frac{u''_{i-1} + 10u''_i + u''_{i+1}}{12}$ by $\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}$ and u_i by v_i we obtain

$$\begin{aligned}\frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + \frac{c_{i-1}v_{i-1} + 10c_iv_i + c_{i+1}v_{i+1}}{12} \\ = \frac{f_{i-1} + 10f_i + f_{i+1}}{12}, \quad i = 2, \dots, n+1,\end{aligned}$$

We modify the first and last equations (i.e. for $i = 2$ and $i = n+1$) with the known values $v_1 = \alpha$ and $v_{n+2} = \beta$ and if $\mathbf{v} =: [v_2, \dots, v_{n+1}]^T$, we deduce that $\mathbf{Av} = \mathbf{b}$ where $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathbf{b} \in \mathbb{R}^n$ are defined by (13.10) and (13.11).

3. numerov1.m

```
n=2;
h=1/(n+1);
x=(0:h:1)';
bf=f1(x)
bc=c1(x)
```

4. numerov2.m

```
numerov1;
A1=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
A2=10*diag(bc(2:n+1))+diag(bc(3:n+1),1)+diag(bc(2:n),-1);
A=A1+h^2/12*A2
```

5. numerov3.m

```
numerov2
b1=bf(1:n)+10*bf(2:n+1)+bf(3:n+2);
b2=zeros(n,1);
b2(1)=(1-h^2/12*bc(1))*alpha;
b2(n)=(1-h^2/12*bc(n+2))*beta;
b=h^2/12*b1+b2
```

6. numerov4.m

```
numerov3
bv=[alpha;A\b;beta];
bu=u1(x);
err=norm(bv-bu,inf)
xx=(0:1/500:1)';
plot(x,bv,'bx--',xx,u1(xx),'r')
```

7. For any polynomial φ of degree less or equal to 5, in (13.9), the bound is 0. So, if u solution of (P) is a such polynomial, its approximant will satisfy $\bar{v} = \bar{u}$ thus $\|\bar{v} - \bar{u}\|_\infty = 0$. An example is given by $u_2(x) = x^4$, $c_2(x) = x$ and $f_2(x) = -12x^2 + x^5$, $\alpha = 0$ and $\beta = 1$.

```
>> numerov5
n =
      5
err =
  5.5511e-17
```

numerov5.m is similar to numerov4.m with c1.m, f1.m, u1.m replaced by c2.m, f2.m, u2.m.

8. numerovorder.m

```
arrn =[5,10,15,20,35,50,75,100];
alpha=1;beta=-1;
for i=1:length(arrn)
    n=arrn(i);
    h=1/(n+1);
    x=(0:h:1)';
```

```

bf=f1(x);
bc=c1(x);
A1=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
A2=10*diag(bc(2:n+1))+diag(bc(3:n+1),1)+diag(bc(2:n),-1);
A=A1+h^2/12*A2;
b1=bf(1:n)+10*bf(2:n+1)+bf(3:n+2);
b2=zeros(n,1);
b2(1)=(1-h^2/12*bc(1))*alpha;
b2(n)=(1-h^2/12*bc(n+2))*beta;
b=h^2/12*b1+b2;
bv=[alpha;A\b;beta];
bu=u1(x);
arerr(i)=norm(bv-bu,inf);
end
plot(log(arrr+1),log(arerr),'x—')
c=polyfit(log(arrr+1),log(arerr),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)');
ylabel('log(error)');

```

The order is 4.

9. numerov6.m

```

n=4
h=1/(n+1);
x=(0:h:1)';
T=0.3; k=4; tau=T/k;
alpha=1; beta=2^((1-sqrt(5))/2);
bc=c3(x);
A1=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
A2=10*diag(bc(2:n+1))+diag(bc(3:n+1),1)+diag(bc(2:n),-1);
A=A1+h^2/12*A2;
B=1/12*(10*eye(n)+diag(ones(n-1,1),1)+diag(ones(n-1,1),-1));
b2=zeros(n,1);
b2(1)=(1-h^2/12*bc(1))*alpha;
b2(n)=(1-h^2/12*bc(n+2))*beta;
w(:,1)=alpha+(beta-alpha)*x(2:n+1);
for j=1:k
    w(:,j+1)=(A+h^2/tau*B)\(h^2/tau*B*w(:,j)+b2);
end;
w(:,k+1)

```

10. numerov7.m

```

n=8;
h=1/(n+1);
x=(0:h:1)';
T=0.3; k=5; tau=T/k;
alpha=1; beta=2^((1-sqrt(5))/2);
bc=c3(x);
A1=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
A2=10*diag(bc(2:n+1))+diag(bc(3:n+1),1)+diag(bc(2:n),-1);
A=A1+h^2/12*A2;

```

```

B=1/12*(10*eye(n)+diag(ones(n-1,1),1)+diag(ones(n-1,1),-1));
b2=zeros(n,1);
b2(1)=(1-h^2/12*bc(1))*alpha;
b2(n)=(1-h^2/12*bc(n+2))*beta;
w(:,1)=alpha+(beta-alpha)*x(2:n+1);
for j=1:k
    w(:,j+1)=(A+h^2/tau*B)\(h^2/tau*B*w(:,j)+b2);
end;
for j=1:k+1
    plot(x,[alpha;w(:,j)];beta,'bx-');
    hold on
end
xx=0:1/500:1;
plot(xx,uinf(xx),'r')
hold off

```

Solutions of Exercise 13.5

1. For $i = 2, \dots, n + 1$, from (13.3) and (13.4), we deduce the approximations

$$u'(t_i) \approx \frac{u(t_{i+1}) - u(t_{i-1})}{2h}, \quad u''(t_i) \approx \frac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2}.$$

Since $-u''(t_i) + rc(t_i)u'(t_i) = f(t_i)$, with the previous approximations, we deduce that

$$-\frac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2} + rc(t_i) \frac{u(t_{i+1}) - u(t_{i-1})}{2h} \approx f(t_i).$$

which leads to the approximation

$$\begin{aligned} & \frac{-v_{i-1} + 2v_i - v_{i+1}}{h^2} + rc(t_i) \frac{v_{i+1} - v_{i-1}}{2h} = f(t_i) \\ \Leftrightarrow & -v_{i-1} + 2v_i - v_{i+1} + \frac{rh}{2}c(t_i)(v_{i+1} - v_{i-1}) = h^2f(t_i) \end{aligned}$$

Now with the boundary conditions $v_1 = u(t_1) = 0 = v_{n+2} = u(t_{n+2})$, the previous equation is a row in $(A + \frac{rh}{2}B)v = f$ and we obtain (13.13) and (13.14).

2. **Programs:** f1.m

```

function y=f1(r,t)
y=r^2*exp(r*t).*(t-1)/(1-exp(r))+r*t;

```

u1.m

```

function y=u1(r,t)
y=t-(1-exp(r*t))/(1-exp(r));

```

c1.m .

```
function y=c1(t)
y=t;
```

3. convdiff.m

```
r=1
n=5
h=1/(n+1);
t=(0:h:1)';
A=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
B=diag(c1(t(2:n+1)))*(diag(ones(n-1,1),1)...
-diag(ones(n-1,1),-1));
f=h^2*f1(r,t(2:n+1));
bv=[0;(A+r*h/2*B)\f;0];
bu=u1(r,t);
error=norm(bv-bu,inf)
tt=0:1/500:1;
plot(t,bv,'---',tt, u1(r,tt))
```

4. cdorder1.m

```
r=1
arrn=100:5:150;
for i=1:length(arrn)
    n=arrn(i);
    h=1/(n+1);
    t=(0:h:1)';
    A=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
    B=diag(c1(t(2:n+1)))*(diag(ones(n-1,1),1)...
    -diag(ones(n-1,1),-1));
    f=h^2*f1(r,t(2:n+1));
    bv=[0;(A+r*h/2*B)\f;0];
    bu=u1(r,t);
    error(i)=norm(bv-bu,inf);
end
plot(log(arrn+1),log(error),'x---')
c=polyfit(log(arrn+1),log(error),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)');
ylabel('log(error)'');
```

5. modifycd.m

```
r=100
n=5
h=1/(n+1);
t=(0:h:1)';
A=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
g=r*h*(c1(t(2:n+1)));
a=(1+cot(g/2))/2-1./g;
B1=diag(c1(t(2:n+1)))*(diag(2*a-1)...
-diag(a(2:n),-1)+diag(1-a(1:n-1),1));
```

```

f=h^2*f1(r,t(2:n+1));
bv=[0;(A+r*h*B1)\f;0]
bu=u1(r,t);
error=norm(bv-bu,inf)
tt=0:1/500:1;
plot(t,bv,'-+',tt,u1(r,tt))

```

6. modifycdorder.m

```

r=100
arrn=40:3:60;
for i=1:length(arrn)
    n=arrn(i);
    h=1/(n+1);
    t=(0:h:1)';
    A=2*eye(n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1);
    g=r*h*(c1(t(2:n+1)));
    a=(1+cotb(g/2))/2-1./g;
    B1=diag(c1(t(2:n+1)))*(diag(2*a-1)...
        -diag(a(2:n),-1)+diag(1-a(1:n-1),1));
    f=h^2*f1(r,t(2:n+1));
    bv=[0;(A+r*h*B1)\f;0];
    bu=u1(r,t);
    error(i)=norm(bv-bu,inf);
end
plot(log(arrn+1),log(error),'-')
c=polyfit(log(arrn+1),log(error),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)');
ylabel('log(error)');

```

13.5.3 Finite Differences in Dimension 2**Solutions of Exercise 13.6**

1. Let $P_{i,j} = (x_i, y_j) \in \omega_h$ and let us write $V_{i,j} = v(P_{i,j})$.

Since $-\Delta_h v(P) + c(P)v(P) = f(P)$ for any $P \in \omega_h$, we deduce that $4V_{i,j} - V_{i-1,j} - V_{i+1,j} - V_{i,j-1} - V_{i,j+1} + h^2 c(x_i, y_j) V_{i,j} = h^2 f(x_i, y_j)$, or

$$(4 + h^2 c(x_i, y_j)) V_{i,j} - V_{i-1,j} - V_{i+1,j} - V_{i,j-1} - V_{i,j+1} = h^2 f(x_i, y_j). \quad (13.30)$$

Now, if one ore more of the points $P_{k,\ell}$ for $(k, \ell) \in \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$ is in $\Omega_h \setminus \omega_h$, then $V(k, \ell) = g(x_k, y_\ell)$, and we move $V(k, \ell)$ to the right hand side of the equation. For example, with the points $P(3h, 4h)$, $P(2h, 3h)$, $P(2h, 2h)$, (13.30) becomes

$$\begin{aligned}(4 + h^2 c(x_3, y_4)) V_{3,4} - V_{2,4} - V_{4,4} - V_{3,3} - V_{3,5} &= h^2 f(x_3, y_4), \\ (4 + h^2 c(x_2, y_3)) V_{2,3} - V_{3,3} - V_{2,2} - V_{2,4} &= h^2 f(x_2, y_3) + g(0, y_3), \\ (4 + h^2 c(x_2, y_2)) V_{2,2} - V_{3,2} - V_{2,3} &= h^2 f(x_2, y_2) + g(0, y_2) + g(x_2, 0).\end{aligned}$$

2. With the MATLAB notations, $\mathbf{V}(:, k) = [V(2, k), \dots, V(n+1, k)]^T$, for $j = 3, \dots, n$, we obtain

$$\mathbf{D}_j \mathbf{V}(:, j) - \mathbf{V}(:, j-1) - \mathbf{V}(:, j+1) = \mathbf{F}(:, j) + \mathbf{G}(:, j)$$

where \mathbf{D}_j is defined in (13.19), $F(i, j) = h^2 f(x_i, y_j)$ and $G(i, j) = 0$ except $G(2, j) = g(0, y_j)$ and $G(n+1, j) = g(1, y_j)$. Similarly, we obtain

$$\begin{aligned}\mathbf{D}_2 \mathbf{V}(:, 2) - \mathbf{V}(:, 3) &= \mathbf{F}(:, 2) + \mathbf{G}(:, 2), \\ \mathbf{D}_{n+1} \mathbf{V}(:, n+1) - \mathbf{V}(:, n) &= \mathbf{F}(:, n+1) + \mathbf{G}(:, n+1),\end{aligned}$$

and we deduce that (Π_h) can be written

$$\mathbf{A} \mathbf{v} = \mathbf{f} + \mathbf{g} \quad (13.31)$$

3. A diagonal element of \mathbf{A} is $4 + c(x_i, y_j)h^2$ for some (i, j) , while on the same row the sum of the non diagonal elements is at most 4. Since $c > 0$, we deduce that \mathbf{A} is strictly diagonally dominant. In, this case (cf. Exercise 2.8), the linear system has a unique solution.

Solutions of Exercise 13.7

- With (13.31) and (13.30), at step k , for $i = 2$ to $n+1$ and $j = 2$ to $n+1$, we obtain that $(4 + h^2 c(x_i, y_j)) V_{i,j}^{k+1} - V_{i-1,j}^{k+1} - V_{i+1,j}^k - V_{i,j-1}^{k+1} - V_{i,j+1}^k = F(i, j)$, which gives $V_{i,j}^{k+1} = \frac{F_{ij} + V_{i-1,j}^{k+1} + V_{i,j-1}^{k+1} + V_{i+1,j}^k + V_{i,j+1}^k}{D_{ij}}$. If we store the computed values $V^k(\cdot)$ and $V^{k+1}(\cdot)$ in $V(\cdot)$, we obtain the proposed algorithm.
- We choose a polynomial solution u with total degree less than 4. In that case, following the comment in Exercise 13.6, the error is equal to zero. For example $u1(x, y) := x^3 + y^2$, $c1(x, y) := x$ and $f1(x, y) = -6x - 4 + x(x^3 + y^2)$

`u1.m`

```
function z=u1(x,y)
z=x.^3+2*y.^2;
```

`c1.m`

```
function z=c1(x,y)
z=x;
```

`f1.m`

```
function z=f1(x,y)
z=-6*x-4+x.* (x.^3+2*y.^2);
```

approxlaplac1.m

```

n=5;
h=1/(n+1);
precis=1e-10;
maxiter=n*20;
[xx,yy]=meshgrid (0:h:1,0:h:1);
xx=xx';yy=yy';
U=u1(xx,yy);
V=zeros(n+2);
V(:,1)=U(:,1);V(:,n+2)=U(:,n+2);
V(1,:)=U(1,:);V(n+2,:)=U(n+2,:);
F=h^2*f1(xx,yy);
D=4+h^2*c1(xx,yy);
iter=0; err=1;
while (iter < maxiter)&(err>precis) % "&" for "and"
    err=0;
    for i=2:n+1
        for j=2:n+1
            z=(F(i,j)+V(i-1,j)+V(i,j-1)+V(i+1,j)...
                +V(i,j+1))/D(i,j);
            err=max(err,abs(z-V(i,j)));
            V(i,j)=z;
        end
    end
    iter=iter+1;
end
if iter==maxiter disp('Non convergence')
else
    error = max(max(abs(U-V)))
end

```

3. u2.m

```

function z=u2(x,y)
z=sin(pi*(2*x+y));

```

c2.m

```

function z=c2(x,y)
z=2*pi^2*ones(length(x));

```

f2.m

```

function z=f2(x,y)
z=7*pi^2*sin(pi*(2*x+y));

```

```

n=10
h=1/(n+1);
precis=1e-10;
maxiter=30*n
[xx,yy]=meshgrid (0:h:1,0:h:1);
xx=xx';yy=yy';
U=u2(xx,yy);

```

```

V=zeros(n+2);
V(:,1)=U(:,1);V(:,n+2)=U(:,n+2);
V(1,:)=U(1,:);V(n+2,:)=U(n+2,:);
F=h^2*f2(xx,yy);
D=4+h^2*c2(xx,yy);
iter=0; err=1;
while (iter < maxiter)&(err>precis)
    err=0;
    for i=2:n+1
        for j=2:n+1
            z=(F(i,j)+V(i-1,j)+V(i,j-1)+V(i+1,j)...
                +V(i,j+1))/D(i,j);
            err=max(err,abs(z-V(i,j)));
            V(i,j)=z;
        end
    end
    iter=iter+1;
end
if iter==maxiter disp('Non convergence')
else
    iter
    error = max(max(abs(U-V)))
    subplot(2,2,1), surf(xx,yy,U)
    axis([0 1 0 1 -1.2 1.2])
    xlabel('x'), ylabel('y')
    title('Exact solution')
    subplot(2,2,2), surf(xx,yy,V)
    xlabel('x'), ylabel('y')
    axis([0 1 0 1 -1.2 1.2])
    title(['Approximation for n= ',num2str(n)])
    subplot(2,2,3), surf(xx,yy,U-V)
    xlabel('x'), ylabel('y')
    title(['Error for n= ',num2str(n)])
    subplot(2,2,4), contour(xx,yy,V-U,10)
    xlabel('x'), ylabel('y')
    title(['Error for n= ',num2str(n)])
end

```

4. Since $\|\bar{U} - \bar{V}\| \times (n+1)^2$ is almost constant, we deduce that the order is 2.
`orderlaplac.m`

```

arrn=60:65;
precis=1e-8;
for k=1:length(arrn)
    n=arrn(k);
    h=1/(n+1);
    maxiter=50*n;
    [xx,yy]=meshgrid(0:h:1,0:h:1);
    xx=xx'; yy=yy';
    U=u2(xx,yy);
    V=zeros(n+2);
    V(:,1)=U(:,1);V(:,n+2)=U(:,n+2);
    V(1,:)=U(1,:);V(n+2,:)=U(n+2,:);
    F=h^2*f2(xx,yy);

```

```

D=4+h^2*c2(xx,yy);
iter=0; err=1;
while (iter < maxiter)&(err>precis)
    err=0;
    for i=2:n+1
        for j=2:n+1
            z=(F(i,j)+V(i-1,j)+V(i,j-1)+V(i+1,j)...
                +V(i,j+1))/D(i,j);
            err=max(err,abs(z-V(i,j)));
            V(i,j)=z;
        end
    end
    iter=iter+1;
end
error(k) = max(max(abs(U-V)));
end
disp('error*(n+1)^2')
error.*(arrn+1).^2

```

Solutions of Exercise 13.8

1. The algorithm:

```

for i = 2 to n + 1
    for j = 2 to n + 1
        
$$z = \frac{F_{ij} + V_{i-1,j} + V_{i,j-1} + V_{i+1,j} + V_{i,j+1} - h^2 \sin(V_{i,j})}{4}$$

        
$$err_{i,j} = |z - V_{i,j}|$$

        
$$V_{i,j} = z$$

    end j
end i

```

2. u3.m

```

function z=u3(x,y)
z=sin(pi*x).*sin(2*pi*y);

```

f3.m

```

function z=f3(x,y)
z=5*pi^2*sin(pi*x).*sin(2*pi*y)+sin(sin(pi*x).*sin(2*pi*y));

```

approxlaplacnonlin.m is similar to approxlaplac2.m with $f2$ and $u2$ replaced by $f3$ and $u3$.

```

...
V=zeros(n+2);
F=h^2*f3(xx,yy);
iter=0; err=1;
while (iter < maxiter)&(err>precis)
    err=0;
    for i=2:n+1
        for j=2:n+1

```

```

z=(F(i ,j)+V(i -1,j)+V(i ,j-1)+V(i +1,j)...
    +V(i ,j+1)-h^2*sin(V(i ,j)))/4;
err=max(err ,abs(z-V(i ,j)));
V(i ,j)=z;
end
end
iter=iter+1;
end
...

```

3. Since $\|\bar{U} - \bar{V}\| \times (n + 1)^2$ is almost constant, we deduce that the order is 2.
`orderlaplacenonlin.m`

```

arrn=60:65;
precis=1e-8;
for k=1:length(arrn)
    n=arrn(k);
    h=1/(n+1);
    maxiter=60*n
    [xx,yy]=meshgrid(0:h:1,0:h:1);
    xx=xx'; yy=yy';
    U=u3(xx,yy);
    V=zeros(n+2);
    F=h^2*f3(xx,yy);
    iter=0; err=1;
    while (iter < maxiter)&(err>precis)
        err=0;
        for i=2:n+1
            for j=2:n+1
                z=(F(i ,j)+V(i -1,j)+V(i ,j-1)+V(i +1,j)...
                    +V(i ,j+1)-h^2*sin(V(i ,j)))/4;
                err=max(err ,abs(z-V(i ,j)));
                V(i ,j)=z;
            end
        end
        iter=iter+1;
    end
    error(k) = max(max(abs(U-V)));
end
disp('error*(n+1)^2')
error.*((arrn+1).^2)

```

13.5.4 The Heat Equation and Approximations

Solution of Exercise 13.9

- Explicit Scheme.** We use formulas (13.21) and (13.23) for the function u at point (x_i, t_ℓ) for $i = 2, \dots, n + 1$ and $\ell = 1, 2, \dots$

$$\begin{aligned} f(x_i, t_\ell) &= \frac{\partial u}{\partial t}(x_i, t_\ell) - \frac{\partial^2 u}{\partial x^2}(x_i, t_\ell) \\ &\approx \frac{u(x_i, t_{\ell+1}) - u(x_i, t_\ell)}{\Delta t} - \frac{u(x_{i-1}, t_\ell) - 2u(x_i, t_\ell) + u(x_{i+1}, t_\ell)}{h^2} \end{aligned}$$

which gives the discrete scheme

$$\Delta t f(x_i, t_\ell) = V(i, \ell+1) - V(i, \ell) - \frac{\Delta t}{h^2} (V(i-1, \ell) - 2V(i, \ell) + V(i+1, \ell))$$

or

$$V(i, \ell+1) = V(i, \ell) + \frac{\Delta t}{h^2} (V(i-1, \ell) - 2V(i, \ell) + V(i+1, \ell)) + \Delta t f(x_i, t_\ell),$$

Since in the first equation (i.e. $i = 2$), $V(1, \ell) = 0$ and also in the last equation (i.e. $i = n+1$), $V(n+2, \ell) = 0$, (13.24) holds.

Implicit Scheme. Similarly, using (13.22) and (13.23) for the approximations of the derivatives of u at point $(x_i, t_{\ell+1})$, we obtain

$$\begin{aligned} \Delta t f(x_i, t_{\ell+1}) &= V(i, \ell+1) - V(i, \ell) \\ &\quad - \frac{\Delta t}{h^2} (V(i-1, \ell+1) - 2V(i, \ell+1) + V(i+1, \ell+1)) \end{aligned}$$

or

$$\begin{aligned} V(i, \ell+1) + \frac{\Delta t}{h^2} (-V(i-1, \ell+1) + 2V(i, \ell+1) - V(i+1, \ell+1)) \\ = V(i, \ell) + \Delta t f(x_i, t_\ell + 1). \end{aligned}$$

so that $(I + \frac{\Delta t}{h^2} A) V(:, \ell+1) = V(:, \ell) + \Delta t F(:, \ell+1)$, holds.

Now the matrix $(I + \frac{\Delta t}{h^2} A)$ is strictly diagonally dominant from which we deduce that the system (13.25) has a unique solution.

Crank-Nicolson scheme. In (13.26), the matrix $(I + \frac{\Delta t}{2h^2} A)$ is strictly diagonally dominant so that $V(:, \ell+1)$ can be uniquely computed from $V(:, \ell)$.

2. From Exercise 13.1 and (13.1) applied to the first derivative of u in t and (13.4) for the second derivative in x^2 , we obtain

$$\begin{aligned} \epsilon_1(x, t) &= \Delta t \frac{\partial u}{\partial t}(x, t) + \frac{1}{2} \Delta t^2 \frac{\partial^2 u}{\partial t^2}(x, t + \alpha t) \\ &\quad - \Delta t \left(\frac{\partial^2 u}{\partial x^2}(x, t) + \frac{h^2}{12} \frac{\partial^4 u}{\partial x^4}(x + \gamma h, t) \right) - \Delta t f(x, t) \\ &= \Delta t \left(\frac{\partial u}{\partial t}(x, t) - \frac{\partial^2 u}{\partial x^2}(x, t) - f(x, t) \right) \\ &\quad + \Delta t \left(\frac{\Delta t}{2} \frac{\partial^2 u}{\partial t^2}(x, t + \alpha t) - \frac{h^2}{12} \frac{\partial^4 u}{\partial x^4}(x + \gamma h, t) \right) \end{aligned}$$

$\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - f = 0$ since u is a solution of (P) and the first part on the right of the equation vanishes. Thus

$$|\epsilon_1(x, t)| \leq \Delta t \left(\frac{\Delta t}{2} M_{t^2}^2 + \frac{h^2}{12} M_{x^4}^4 \right),$$

where $M_{t^2}^2 := \max_{(p,q) \in R} \left| \frac{\partial^2 u(p, q)}{\partial t^2} \right|$ and $M_{x^4}^4 := \max_{(p,q) \in R} \left| \frac{\partial^4 u(p, q)}{\partial x^4} \right|$, and $R := [x - h, x + h] \times [t, t + \Delta t]$. Similarly using (13.2) and (13.4) at $(x, t + \Delta t)$, $|\epsilon_2(x, t)| \leq \Delta t \left(\frac{\Delta t}{2} M_{t^2}^2 + \frac{h^2}{12} M_{x^4}^4 \right)$ and the order is also 1.

For the Crank-Nicolson scheme we use a Taylor expansion at (x, t) then at $(x, t + \Delta t)$

$$\begin{aligned} u(x, t + \Delta t) - u(x, t) &= \Delta t \frac{\partial u}{\partial t}(x, t) + \frac{(\Delta t)^2}{2} \frac{\partial^2 u}{\partial t^2}(x, t) \\ &\quad + \frac{(\Delta t)^3}{6} \frac{\partial^3 u}{\partial t^3}(x, t + \alpha_1 \Delta t) \\ u(x, t + \Delta t) - u(x, t) &= \Delta t \frac{\partial u}{\partial t}(x, t + \Delta t) - \frac{1}{2} (\Delta t)^2 \frac{\partial^2 u}{\partial t^2}(x, t + \Delta t) \\ &\quad + \frac{1}{6} (\Delta t)^3 \frac{\partial^3 u}{\partial t^3}(x, t + \alpha_2 \Delta t) \end{aligned}$$

and we compute the average of this two equations to obtain the first part $\epsilon_3(x, t)$: $u(x, t + \Delta t) - u(x, t)$.

Then we use (13.4) at (x, t) and also at $(x, t + \Delta t)$. There exists two values of γ , γ_1 for t and γ_2 for $t + \Delta t$, such that:

$$u(x - h, \cdot) - 2u(x, \cdot) + u(x + h, \cdot) = h^2 \frac{\partial^2 u}{\partial x^2}(x, \cdot) + \frac{h^4}{12} \frac{\partial^4 u}{\partial x^4}(x + \gamma h, \cdot).$$

We deduce

$$\begin{aligned} \epsilon_3(x, t) &= \frac{\Delta t}{2} \frac{\partial u}{\partial t}(x, t) + \frac{\Delta t^2}{4} \frac{\partial^2 u}{\partial t^2}(x, t) + \frac{\Delta t}{2} \frac{\partial u}{\partial t}(x, t + \Delta t) \\ &\quad - \frac{\Delta t^2}{4} \frac{\partial^2 u}{\partial t^2}(x, t + \Delta t) \\ &\quad + \frac{\Delta t^3}{12} \left(\frac{\partial^3 u}{\partial t^3}(x, t + \alpha_1 \Delta t) + \frac{\partial^3 u}{\partial t^3}(x, t + \alpha_2 \Delta t) \right) \\ &\quad - \frac{\Delta t}{2} \left(\frac{\partial^2 u}{\partial x^2}(x, t) + \frac{\partial^2 u}{\partial x^2}(x, t + \Delta t) \right) \\ &\quad - \frac{\Delta t h^2}{24} \left(\frac{\partial^4 u}{\partial x^4}(x + \gamma_1 h, t) + \frac{\partial^4 u}{\partial x^4}(x + \gamma_2 h, t + \Delta t) \right) \\ &\quad - \frac{\Delta t}{2} (f(x, t) + f(x, t + \Delta t)). \end{aligned}$$

Since $\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} - f = 0$ at (x, t) and $(x, t + \Delta t)$, and

$\frac{\partial^2 u}{\partial t^2}(x, t) - \frac{\partial^2 u}{\partial t^2}(x, t + \Delta t) = -\Delta t \frac{\partial^3 u}{\partial t^3}(x, t + \alpha_3 \Delta t)$, we deduce that

$$|\epsilon_3(x, t)| \leq (\Delta t)^3 \frac{5}{12} M_{t^3}^3 + \Delta t \frac{1}{12} M_{x^4}^4 h^2.$$

3. If $h = O(\Delta t)$, then $|\epsilon_1(x, t)| \leq C_1(\Delta t)^2$. In this case, the order is 1 and we also have order 1 for the implicit scheme. Similarly, if $f = O(\Delta t)$, we obtain $|\epsilon_3(x, t)| \leq C_3(\Delta t)^3$ which gives order 2.
4. a. If $u(x, t) = \sin(p\pi x)e^{-p^2\pi^2 t}$, then $\frac{\partial u}{\partial t} - \frac{\partial^2 u}{\partial x^2} = 0$. Moreover, $u(x, 0) = \sin(p\pi x) = u_0(x)$ and $u(0, t) = u(1, t) = 0$, so that u is the solution of (P) .
- b. In every scheme $V(i, 0) = \sin(p\pi x_i) = \sin(p\pi hi)$. Then we can prove inductively that

explicit scheme: $V_1(i, n) = \alpha_1 V(i, n - 1)$ where $\alpha_1 = 1 - \zeta$

implicit scheme: $V_2(i, n) = \alpha_2 V(i, n - 1)$ where $\alpha_2 = \frac{1}{1 + \zeta}$

Crank-N. scheme: $V_3(i, n) = \alpha_3 V(i, n - 1)$ where $\alpha_3 = \frac{1 - \zeta/2}{1 + \zeta/2}$,

where $\zeta = 4 \frac{\Delta t}{h^2} \sin^2(p\pi h/2) \geq 0$. In every case, we have a geometric sequence.

- c. The discrete solution is bounded when n tends to $+\infty$ as soon as $|\alpha_i| \leq 1$. But $\alpha_1 < 1$ and $\alpha_2, \alpha_3 \geq -1$. The implicit and Crank-Nicolson schemes are unconditionally stable. On the contrary, $-1 \leq \alpha_1 \Leftrightarrow 1/2 \geq \frac{\Delta t}{h^2} \sin^2(p\pi h/2)$ and for example for $p = n + 1$, the stability condition for the explicit scheme is $1/2 \geq \Delta t/h^2$.

② **Comment:** The stability condition $\Delta t \leq h^2/2$ shows that we generally must use very small time steps in the explicit scheme. This can severely restrict the utility of this scheme. ☺

Solution of Exercise 13.10

1. `u0.m`

```
function y=u0(x)
y=sin(pi*x);
```

`u.m`

```
function y=u(x,t)
y=sin(pi*x).*exp(-pi^2*t);
```

`heq1.m`

```
n=3;h=1/(n+1);
T=0.4;
x=(0:h:1)';
V(:,1)=u0(x(2:n+1));
A=2*diag(ones(n,1))-diag(ones(n-1,1),-1)-diag(ones(n-1,1),1)
```

2. heq2.m

```

clear
n=6;h=1/(n+1);x=(0:h:1)';
T=0.4;dt=0.1;t=0:dt:T;
V(:,1)=u0(x(2:n+1));
A=2*diag(ones(n,1)) - diag(ones(n-1,1),-1) - diag(ones(n-1,1),1);
M=eye(n)+dt/h^2*A;
for l=1:length(t)-1
    V(:,l+1)=M\V(:,l);
end
bV=zeros(n+2,length(t));
bV(2:n+1,:)=V;
[xx,tt]=meshgrid(x,t);
bU=u(xx,tt);
subplot(221)
surf(xx,tt,bU); title('Exact solution')
subplot(222)
surf(xx,tt,bV'); title('Approximate solution')
subplot(223)
surf(xx,tt,bV'-bU)
title(['Error for n = ',int2str(n)])
subplot(224)
contour(xx,tt,bV')
title('Level curves for the approx. sol.')

```

3. heq3.m

```

n=200;h=1/(n+1);x=(0:h:1)';
T=0.4;
V(:,1)=u0(x(2:n+1));
A=2*diag(ones(n,1)) - diag(ones(n-1,1),-1) - diag(ones(n-1,1),1);
arrdt=[0.01,0.005,0.002,0.001];
for i=1:length(arrdt)
    dt=arrdt(i);
    t=0:dt:T;
    M=eye(n)+dt/h^2*A;
    for l=1:length(t)-1
        V(:,l+1)=M\V(:,l);
    end
    bV=zeros(n+2,length(t));
    bV(2:n+1,:)=V;
    [xx,tt]=meshgrid(x,t);
    bU=u(xx,tt);
    err(i)=max(max(abs(bV'-bU)));
end
plot(log(arrdt),log(err),'x—')
c=polyfit(log(arrdt),log(err),1);
q=c(1);
title(['Slope of the regression line: ',num2str(q)]);
xlabel('log(dt)'); ylabel('log(error)');

```

4. heq4.m

```

T=0.00005; dt=0.000001; t=0:dt:T;
arrn=10:20;
for i=1:length(arrn)
    n=arrn(i); h=1/(n+1); x=(0:h:1)';
    V(:,1)=u0(x(2:n+1));
    A=2*diag(ones(n,1))-diag(ones(n-1,1),-1)...
        -diag(ones(n-1,1),1);
    M=eye(n)+dt/h^2*A;
    for l=1:length(t)-1
        V(:,l+1)=M\V(:,l);
    end
    bV=zeros(n+2,length(t));
    bV(2:n+1,:)=V;
    [xx,tt]=meshgrid(x,t);
    bU=u(xx,tt);
    err(i)=max(max(abs(bV'-bU)));
    clear V
end
plot(log(arrn+1),log(err),'x—')
c=polyfit(log(arrn+1),log(err),1);
q=c(1);
title([ 'Slope of the regression line: ',num2str(q)]);
xlabel('log(n+1)'); ylabel('log(error)');

```

5. heq5.m and heq6.m. Change the central parts of heq3.m and heq4.m into

```

M=eye(n)+dt/(2*h^2)*A;
N=eye(n)-dt/(2*h^2)*A;
for l=1:length(t)-1
    V(:,l+1)=M\N\V(:,l));
end

```

References

1. J. Bastien, *Introduction à l'analyse numérique : Applications sous MATLAB*, Dunod, 2003.
2. A. Biran, M. Breiner, *MATLAB 6 for Engineers*, Prentice Hall, 2002, 3th ed.
3. A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, 1996.
4. K. Chen, P. Giblin, A. Irving, *Mathematical Explorations with MATLAB*, Cambridge Univ. Press, 1999.
5. E. Cohen, R. F. Riesenfeld, G. Elber, *Geometric Modeling with Splines: An Introduction*, A.K. Peters, Ltd., 2001.
6. J. Cooper, *MATLAB Companion for Multivariable Calculus (a)*, Academic Press, 2001.
7. M. Crouzeix, A. Mignot, *Analyse numérique des équations différentielles*, Masson, 1989.
8. C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, 2001, Rev. Ed.
9. P. Deuflhard, A. Hohmann, *Numerical Analysis in Modern Scientific Computing, An Introduction*, Springer-Verlag, 2003, 2nd ed.
10. A. Fortin, *Analyse numérique pour ingénieurs*, Ed. de l'Ecole Polytechnique de Montréal, 1995.
11. W. Gander, J. Hřebíček, *Solving Problems in Scientific Computing using MAPPLE and MATLAB*, Springer-Verlag, 2004, 4th ed.
12. S. Godounov, V. Riabenki, *Schémas aux différences*, MIR, 1977.
13. G.H. Golub, C.F. Van Loan, *Matrix Computations*, The Johns Hopkins Univ. Press, 1996, 3rd Ed.
14. A. Greenbaum, *Iterative mMethods for Solving Linear Systems*, SIAM, 1997.
15. F. Gustafsson, N. Bergman, *MATLAB for Engineers Explained*, Springer-Verlag, 2003.
16. D.J. Higham, N. Higham, *MATLAB Guide*, SIAM, 2005, 2nd Ed.
17. A. Kharab, R. B. Guenther, *Introduction to Numerical Methods (a) : a MATLAB Approach*, Chapman and Hall, 2002.
18. P. Lascaux and R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, vol 1 and 2, Dunod, 2004.
19. A.J. Laub, *Matrix Analysis for Scientists and Engineers*, SIAM, 2005.
20. P. Linz, R. Wang, *Exploring Numerical Methods: An Introduction to Scientific Computing using MATLAB*, Jones and Barlett Pub., 2003.
21. D. Marsh, *Applied Geometry for Computer Graphics and CAD*, Springer-Verlag, 1999.
22. C. B. Moler, *Numerical Computing with MATLAB*, SIAM, 2004.
23. G.M. Phillips, *Interpolation and Approximation by Polynomials*, CMS Books in Mathematics, Springer-Verlag 2003.
24. A. Quarteroni, R. Sacco, F. Saleri *Numerical Mathematics*, Springer-Verlag, 2000.
25. A. Quarteroni, R. Sacco, F. Saleri *Scientific Computing with MATLAB*, Springer-Verlag, 2003.
26. D. Salomon, *Curves and Surfaces for Computer Graphics*, Springer-Verlag, 2006.

27. M. Schatzman, *Numerical Analysis: A Mathematical Introduction*, Oxford Univ. Press, 2002.
28. B.J. Schroers, *Ordinary Differential Equations: a Practical Guide*, Cambridge Univ. Press, 2011.
29. E. Süli, D. Mayers, *An introduction to Numerical Analysis*, Cambridge Univ. Press, 2003.
30. L.N. Trefethen & D. Bau III, *Numerical Linear Algebra*, SIAM, 1997.
31. C.F. Van Loan, K.-Y. Daysy Fan, *Insight Through Computing*, SIAM, 2010.
32. H. B. Wilson, L. H. Turcotte, D. Halpern, *Advanced Mathematics and Mechanics Applications using MATLAB*, Chapman and Hall, 2003, 3rd ed.
33. K. Yosida, *Functional Analysis*, Springer-Verlag, 1980, 6-th ed.

Index of Names

- Bézier, P., 131, 133
Bernstein, S.N., 106, 131, 136, 154, 249, 252
Boole, G., 180

Cauchy, A.L., 282
Chebyshev, P.L., 105, 111, 250
Cholesky, A.L., 9, 16
Cotes, R., 179
Crank, N., 338

de Casteljau, P., 131, 134, 253
Deslauriers, G., 76
Dirichlet, J.P.G.L., 259
Dubuc, S., 76
Dyn, N., 76

Euler, L., 192, 283

Fourier, 223
Fourier, J., 258

Gauss, J.C.F., 70, 182, 334
Gershgorin, S.A., 26, 159
Golay, M.J.E., 232
Gregory, J., 76

Haar, A., 261
Henon, M., 68
Hermite, C., 103, 104, 153
Heun, K., 283
Hilbert, D., 48

Jacobi, C.G.J., 69
Jordan, C., 27

Kutta, M.W., 283

Lagrange, J.L., 103, 104, 178, 179, 250
Laplace, P.S., 322
Legendre, A.M., 182
Levin, D., 76
Lipschitz, R.O.S., 282
Love, E.R., 188

Maclaurin, C., 192

Newton, I., 66, 105, 107, 179
Nicolson, P., 338
Numerov, B.V., 324

Pythagoras, 224

Rayleigh, J.W.S., 25, 29
Remez, E.Y., 256
Rolle, M., 123
Romberg, W., 193
Runge, C.D.T., 110, 283

Savitzky, A., 233
Seidel, P.L.V., 70, 334
Simpson, T., 166, 179

Taylor, B., 94, 110, 306, 342

Vandermonde, A.T., 46, 235, 255

Weierstrass, K.T.W., 251

Young, T., 323

Subject Index

- Backward Euler method, 283
Basic QR algorithm, 31
Basis, 104, 105, 136, 234
 Bernstein, 106
 Hermite, 106
 Lagrange, 105, 107, 178
 Newton, 105, 107
 power, 105
Bending moment, 323
Bernstein
 basis, 106, 136, 155
 form, 154
 polynomials, 136, 252
Bézier
 coefficients, 136
 curve, 132
 end point property, 133
 point matrix, 132
 points, 132
 tangent end point, 138
Boole's rule, 180
Cauchy problem, 282
Characteristic polynomial, 25
Chebyshev
 approximation, 250
 polynomials, 250
 sites, 105, 111, 117, 251
Condition number, 44
Conic, 226
Consistency error, 284
Consistent scheme, 284
Control polygon, 131
Convergent scheme, 284
Crank-Nicolson scheme, 338, 358
Cubic, 153
 convergence, 30
 spline, 153, 157, 190
Curve
 Bézier curve, 132
 control curve, 132
 parametric curve, 132, 139
 plane curve, 132
 smooth curve, 141
 space curve, 132
Damped pendulum, 80
de Casteljau algorithm, 131, 132, 253
Degree of precision, 177, 178
Determinant, 26
Differential equation, 236
 eigenfunctions, 32
 eigenvalue, 32
Divided difference, 108
Dominant eigenvalue, 28
Eigenfunctions, 32
Eigenspace, 25
Eigenvalue, 25, 286
 dominant, 28
Eigenvector, 25
Equioscillation, 249
Euler's method, 283
 backward, 285, 287
 direct, 285, 287
Even function, 259
Explicit scheme, 338, 357
Fixed-point, 66
Forward differences, 136

- Fourier sine series, 259
- Fourth order Runge-Kutta, 283
- Gauss-Legendre rule, 181, 182
- Gauss-Seidel method, 70, 334
- Generalized inverse, 225
- Gershgorin's circle theorem, 159
- Heat, 260, 337
- Heat conductivity, 337
- Hermite
 - basis, 106
 - cubic, 153
 - interpolant, 154
 - interpolation, 103–105
 - quadratic, 153, 154
- Hermitian matrix, 26
- Heun's method, 283, 288
- Hilbert matrix, 48
- Homogeneous system, 286
- Implicit scheme, 328, 338, 358
- Initial condition, 282, 285
- Inner product, 183
- Integral equation, 188, 196
- Interpolation sites, 104, 179
- Inverse power method, 28
- Iterative method, 79
- Jacobi's method, 69
- Jordan block, 27
- Kernel, 188
- Knot, 154
- Lagrange
 - basis, 105, 107, 178
 - error, 250
 - form, 105, 107
 - interpolant, 177, 179
 - interpolation, 103
 - polynomial, 179, 250
- Least squares problem, 221
- Least squares solution, 221
- Length of a curve, 153
- Linear regression, 187
- Linear system
 - homogenous, 10
 - overdetermined, 10
 - square, 10
 - underdetermined, 10
- Lipschitz condition, 282, 288
- Love's equation, 188
- Matrix
 - block, 12
 - block multiplication, 12
 - characteristic polynomial, 25
 - Cholesky factorization, 16
 - column, 13
 - column space, 10
 - conjugate transpose, 26
 - determinant, 10
 - diagonal, 27
 - eigenvalue, 25
 - eigenvector, 25
 - Hermitian, 26
 - inverse, 10, 13
 - Jordan block, 27
 - nonsingular, 10, 13, 21
 - norm, 43
 - nullity, 10
 - null space, 10
 - positive, 324
 - positive definite, 16
 - rank, 10, 226
 - similar, 26
 - spectral norm, 44
 - strictly diagonally dominant, 13
 - symmetric, 16, 26
 - triangular, 13, 21
 - unitary, 26
 - unitary similarity transformation, 26
 - Vandermonde, 46
- Mean value Theorem, 342
- Mesh, 337
- Mid-point rule, 179, 180
- Minimum, 52
- Monte Carlo method, 184
- Multiplicity, 25
- Newton basis, 105, 107
- Newton form
 - quadratic, 105
- Newton-Cotes formula, 179
- Node polynomial, 178
- Nodes, 178
- Noisy signal, 229
- Norm, 43
 - consistency condition, 43
 - infinity-norm, 44
 - one-norm, 44
 - two-norm, 44
- Normal equations, 222
- Numerov scheme, 324
- Odd function, 259
- Operator norm, 43

- Order, 284, 339
Orthogonal decomposition, 223
Orthogonal polynomials, 182
Orthogonal projection, 222
Orthonormal polynomials, 182
Oscillations of a pendulum, 78
- Partition, 282
Periodic signal, 229
Piecewise Bernstein form, 155
Piecewise polynomials, 153
Positive matrix, 324
Positive vector, 324
Power basis, 105
- QR factorization, 30
QR-method, 25
Quadratic, 153
convergence, 30
Hermite interpolant, 154
- Quadrature, 179
Quadrature rule, 178
- Rayleigh quotient, 25, 28
Regression, 226
Regression line, 187
Romberg's method, 193
Rule
Boole, 180
Gauss-Legendre, 181
mid-point, 180
Simpson, 180, 181
Simpson 3/8, 180
trapezoidal, 180
- Runge phenomenon, 110
- Sampled points, 184
Savitsky-Golay Filter, 232
Secant method, 299
- Shift, 28
Similar matrices, 26
Simpson 3/8 rule, 180
Simpson's rule, 179, 180
Sparse matrix, 334
Specific heat, 337
Spectral norm, 44
Spectral radius, 26, 44, 71
Spectrum, 25
Stable scheme, 284
Stationary solution, 328
Subdivision scheme, 76
Summation by parts, 138
Symmetric matrix, 26
- Taylor, 94
Taylor expansion, 306
Temperature, 337
Trace, 25
Trapezoidal method, 283
Trapezoid/Trapezoidal rule, 179, 180
Truncation error, 178
- Uniform norm, 249
Uniform sites, 105, 117
Unitary matrix, 26
Unitary similarity transformation, 26
- Vandermonde, 235
Vandermonde matrix, 46, 246
Vector
norm, 43
positive, 324
- Weierstrass Theorem, 249, 251
Weight function, 178, 182
Weights, 178
- Young's modulus, 323

MATLAB Index

|, 207
*, 4
.* , 4, 227, 246
.J, 126
. ., 63
==, 84
&, 207

abs, 5, 28
and, 354
axis, 211, 244, 355

case, 274
cat, 134
close all, 6
comet, 6
cond, 57, 58
contour, 7, 228, 340, 355

det, 4, 17, 56
diag, 2, 14, 21, 60, 126, 173, 348
disp, 2, 18, 58, 84, 206, 210, 242, 356

eig, 4, 31, 61
else, 84, 274, 354
eps, 6
eye, 2, 60, 126, 348

feval, 53, 63, 209, 273
figure, 6
find, 274
fontsize, 121
format, 2

ginput, 144
grid on, 318

if, 84, 274, 354
input, 2
int2str, 18
inv, 4, 18, 58, 246

length, 57, 126, 210, 246, 318

max, 5, 28, 52, 61, 62, 273
mesh, 7
meshgrid, 103, 126, 127, 228, 335, 340, 354
M-file, 4
M-function, 4
min, 61, 62

ndgrid, 134
nnz, 206, 207
norm, 56, 58, 61, 211
num2str, 84, 210

ode, 282
ones, 2, 14, 21, 58, 60–63, 126, 173, 210, 246
otherwise, 274

pause, 170
plot, 6, 61, 63, 84, 121
plot3, 135
plottools, 7
polyfit, 57, 84, 110, 125, 210, 226, 242
polyval, 53, 63, 110, 125, 242, 273
prod, 126, 200

qr, 31
quad, 177
quadl, 53, 177, 204

rand, 2, 61, 206, 207, 230
rank, 4, 10

sin, 244
sort, 230
spline, 161, 211
sqrt, 2, 61, 204
squeeze, 134
subplot, 84, 121, 211, 242, 244, 355
sum, 2, 126, 209
surf, 7, 127, 340, 355
switch, 274
tic,toc, 5
title, 121, 244
uicontrol, 170
while, 84, 144, 170, 354
xlabel, 210
zeros, 3, 246, 318