An INTRODUCTION to NUMERICAL METHODS

A MATLAB® Approach

ABDELWAHAB KHARAB
King Fahd University of Petroleum and Minerals

RONALD B. GUENTHER
Oregon State University

CHAPMAN & HALL/CRC

A CRC Press Company
Boca Raton London New York Washington, D.C.

Library of Congress Cataloging-in-Publication Data

Kharab, Abdelwahab.
An introduction to numerical methods : a MATLAB approach / Abdelwahab Kharab and Ronald B. Guenther.
p. cm.
Includes bibliographical references and index.
ISBN 1-58488-281-6 (alk. paper)
1. Numerical analysis—Data processing. 2. MATLAB. I. Guenther, Ronald B. II Title.

QA297 .K52 2001
519.4'0285—dc21 2001047074

This book contains information obtained from authentic and highly regarded sources. Reprinted material is quoted with permission, and sources are indicated. A wide variety of references are listed. Reasonable efforts have been made to publish reliable data and information, but the author and the publisher cannot assume responsibility for the validity of all materials or for the consequences of their use.

Neither this book nor any part may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, microfilming, and recording, or by any information storage or retrieval system, without prior permission in writing from the publisher.

The consent of CRC Press LLC does not extend to copying for general distribution, for promotion, for creating new works, or for resale. Specific permission must be obtained in writing from CRC Press LLC for such copying.

Direct all inquiries to CRC Press LLC, 2000 N.W. Corporate Blvd., Boca Raton, Florida 33431.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation, without intent to infringe.

Visit the CRC Press Web site at www.crcpress.com

© 2002 by Chapman & Hall/CRC

No claim to original U.S. Government works
International Standard Book Number 1-58488-281-6
Library of Congress Card Number 2001047074
Printed in the United States of America 1 2 3 4 5 6 7 8 9 0
Printed on acid-free paper

Preface

This is a textbook designed for an introductory course in numerical analysis. It grew out of a set of lecture notes from a course on numerical analysis that has been continuously offered at the departments of mathematics of the King Fahd University of Petroleum and Minerals and Oregon State University. It deals with the theory and application of the most commonly used numerical methods for solving numerical problems on microcomputers. It is intended for students in mathematics, science, and engineering who have completed the introductory calculus sequence. In addition, the reader is assumed to have taken a structured programming course. The thrust of this text is to assist the students to become familiar with the most common numerical methods encountered in science and engineering. The content material of this book has been designed to be compatible with any introductory numerical textbook that exists in the market. Students will be able to examine and solve many numerical problems, using MATLAB^{TM}[1] in a short period of time.

Due to the rapid advancement of computer technology and software developments, we have used MATLAB as the computing environment throughout all the chapters of the book. Each numerical method discussed in this book is demonstrated through the use of MATLAB which is easy to use and has many features such as:

[1] MATLAB^{TM} is a registered tradmark of the MathWorks, Inc.
For product information, please contact:
The MathWorks, Inc.
3 Apple Hill Drive
Natick, MA 01760-2098 USA
Tel: 508-647-7000 Fax: 508-647-7001
E-mail: info@mathworks.com Web: www.mathworks.com

1. Powerful matrix structure,

2. Powerful two- and three-dimensional graphing facilities,

3. A vast number of powerful built-in functions,

4. MATLAB's structured programming style that resembles FORTRAN and BASIC.

This book introduces students to a wide range of useful and important algorithms. Computer results are presented with full details so that the main steps of the algorithm of each numerical method are visualized and interpreted. For this reason, a supplement CD-ROM attached at the back of the book, has been developed for students' use with MATLAB. The CD-ROM contains simple MATLAB functions that give a clear step-by-step explanation of the mechanism behind the algorithm of each numerical method covered. Emphasis is placed on understanding how the methods work. These functions guide the student through the calculations necessary to understand the algorithm(s). The main feature of this book beside the use of MATLAB as its computing environment, is that the style of the book is easy, direct and simple. This is likely to boost students' confidence in their ability to master the elements of the subject.

The book is organized in a fairly standard manner. Topics that are simpler, both theoretically and computationally, come first; for example root of equations is covered in Chapter 3. Chapter 2 contains an introduction to computer floating point arithmetic, errors, and interval arithmetic.

Both direct and iterative methods are presented in Chapter 4 for solving systems of linear equations.

Interpolation, spline functions, and concepts of least squares data fitting are the subjects of Chapters 5, 6, and 7. Interpolation forms the theoretical basis for much of numerical analysis.

Chapter 8 is devoted to numerical differentiation and integration. Several efficient integration techniques are presented.

In Chapter 9, a wide variety of numerical techniques is presented for solving ordinary differential equations. An introduction for solving boundary value problems is presented in Chapter 10. In Chapter 11, the book presents a new numerical method (Quadratic method) developed by Dr. Elgindi and Dr. Kharab that approximates the eigenpairs of a matrix.

The last chapter provides a basic introduction to numerical techniques for solving partial differential equations.

It is the purpose of this book to implement various important numerical methods on a microcomputer and to provide not only a source of theoretical information on the methods covered, but also to allow the student to easily interact with the microcomputer and the algorithm for each method using MATLAB.

This text should provide an excellent tool suitable for teaching numerical method courses at colleges and universities. It is also suitable for self-study purposes.

Finally, we wish to thank the many persons who have helped us in the creation of this book. They are: R. Baker Kearfoot of the University of Southwestern Louisiana, Rachid Kharab of the University of Rouen, A. Laradji of the King Fahd University of Petroleum & Minerals, and L. Cheded of the King Fahd University of Petroleum & Minerals, who encouraged us to submit a book proposal.

Recommendations that were provided to us by the publisher were used to improve the book. We acknowledge the reviewers for these recommendations.

The authors remain very grateful to the editorial and production staff of Chapman & Hall/CRC who have been helpful and available at all stages. Among them are Samar Haddad, project Editor, who was in constant communication with me during the preparation of the final version of the book and Sunil Nair, mathematics and statistics publisher, for his assistance and encouragement.

Finally, the authors are grateful for the financial and facilities support provided by King Fahd University of Petroleum & Minerals, Dhahran, Saudi Arabia.

Suggestions for improvements to the book are always welcome and can be made by e-mail at: awkharab@kfupm.edu.sa.

Abdelwahab Kharab
Ronald B. Guenther

Contents

1 Introduction **1**

1.1 ABOUT THE SOFTWARE MATLAB 1

1.2 AN INTRODUCTION TO MATLAB 2

1.2.1 Matrices and matrix computation 2

1.2.2 Output format 6

1.2.3 Planar plots 7

1.2.4 3-D mesh plots 8

1.2.5 Function files 9

1.2.6 Defining functions 10

1.2.7 Relations and loops 11

1.3 TAYLOR SERIES 13

2 Number System and Errors **21**

2.1 FLOATING-POINT ARITHMETIC 21

2.2 ROUND-OFF ERRORS 26

2.3 TRUNCATION ERROR 29

2.4 INTERVAL ARITHMETIC 30

3 Roots of Equations **37**

3.1 THE BISECTION METHOD 37

3.2 THE METHOD OF FALSE POSITION 44

3.3 THE SECANT METHOD 48

3.4 NEWTON'S METHOD 52

3.5 CONVERGENCE OF THE NEWTON AND SECANT METHODS 58

3.6 FIXED POINT ITERATION 60

3.7 MULTIPLE ROOTS AND THE MODIFIED NEWTON METHOD 66

3.8 NEWTON'S METHOD FOR NONLINEAR SYSTEMS 72

4 System of Linear Equations 79
4.1 MATRICES AND MATRIX OPERATIONS 79
4.2 NAIVE GAUSSIAN ELIMINATION 83
4.3 GAUSSIAN ELIMINATION WITH SCALED PARTIAL PIVOTING 92
4.4 LU DECOMPOSITION 102
4.4.1 Crout's and Choleski's methods 102
4.4.2 Gaussian elimination method 108
4.5 ITERATIVE METHODS 115
4.5.1 Jacobi iterative method 115
4.5.2 Gauss-Seidel iterative method 119
4.5.3 Convergence 123

5 Interpolation 131
5.1 POLYNOMIAL INTERPOLATION THEORY 132
5.2 NEWTON'S DIVIDED DIFFERENCE INTERPOLATING POLYNOMIAL 134
5.3 THE ERROR OF THE INTERPOLATING POLYNOMIAL 144
5.4 LAGRANGE INTERPOLATING POLYNOMIAL . . . 147

6 Interpolation with Spline Functions 153
6.1 PIECEWISE LINEAR INTERPOLATION 153
6.2 QUADRATIC SPLINE 158
6.3 NATURAL CUBIC SPLINES 161

7 The Method of Least Squares 175
7.1 LINEAR LEAST SQUARES 176
7.2 LEAST SQUARES POLYNOMIAL 182
7.3 NONLINEAR LEAST SQUARES 190
7.3.1 Exponential form 191
7.3.2 Hyperbolic form 193
7.4 TRIGONOMETRIC LEAST SQUARES POLYNOMIAL 197

8 Numerical Differentiation and Integration 203
8.1 NUMERICAL DIFFERENTIATION 204
8.2 RICHARDSON'S FORMULA 208

8.3 TRAPEZOIDAL RULE 214
8.4 SIMPSON'S RULE 222
8.5 ROMBERG ALGORITHM 230
8.6 GAUSSIAN QUADRATURE 237

9 Numerical Methods for Ordinary Differential Equations 247
9.1 EULER'S METHOD 247
9.2 ERROR ANALYSIS 254
9.3 HIGHER ORDER TAYLOR SERIES METHODS . . . 259
9.4 RUNGE-KUTTA METHODS 262
9.5 MULTISTEP METHODS 272
9.6 ADAMS-BASHFORTH METHOD 273
9.7 PREDICTOR-CORRECTOR METHODS 282
9.8 ADAMS-MOULTON METHOD 283
9.9 NUMERICAL STABILITY 289
9.10 HIGHER ORDER EQUATIONS AND SYSTEMS . . . 292
9.11 IMPLICIT METHODS AND STIFF SYSTEMS 299
9.12 PHASE PLANE ANALYSIS: CHAOTIC EQUATIONS 301

10 Boundary-Value Problems 311
10.1 FINITE-DIFFERENCE METHODS 311
10.2 SHOOTING METHODS 318
10.2.1 The nonlinear case 318
10.2.2 The linear case 321

11 Eigenvalues and Eigenvectors 329
11.1 BASIC THEORY 329
11.2 THE POWER METHOD 334
11.3 THE QUADRATIC METHOD 338
11.4 EIGENVALUES FOR BOUNDARY-VALUE PROBLEMS 349
11.5 BIFURCATIONS IN DIFFERENTIAL EQUATIONS . 352

12 Partial Differential Equations 359
12.1 PARABOLIC EQUATIONS 360
12.1.1 Explicit methods 360
12.1.2 Implicit methods 366
12.2 HYPERBOLIC EQUATIONS 373
12.3 ELLIPTIC EQUATIONS 380

Bibliography and References **389**

Appendix A Calculus Review **393**
A.0.1 Limits and continuity 393
A.0.2 Differentiation 394
A.0.3 Integration 394

Appendix B MATLAB Built-In Functions **397**

Appendix C Text MATLAB Functions **401**

Answers to Selected Exercises **403**

Index **427**

Chapter 1

Introduction

The Taylor Series is one of the most important tools in numerical analysis. It constitutes the foundation of numerical methods and will be used in most of the chapters of this text. From the Taylor Series, we can derive the formulas and error estimates of the many numerical techniques used. This chapter contains a review of the Taylor Series, and a brief introduction to MATLAB.

1.1 ABOUT THE SOFTWARE MATLAB

The widespread availability of personal computers, their low cost, utility, and convenience, is rapidly becoming preferable to the large, mainframe computers. It gives an opportunity to students to more effectively integrate computing in the classroom. To exploit this opportunity, students in science and engineering should make use of the software available for implementing numerical methods on microcomputers. An Example of these tools is the symbolic algebra programs such as MATLAB, MAPLE, DERIVE, MathCad, and Mathematica. Fortunately, the power and computing capabilities of these software programs have not been fully used in classrooms by teachers in science and engineering. For this reason, part of this text is intended to introduce the computational power of MATLAB to modern numerical methods.

MATLAB has several advantages. There are three major elements that have contributed to its immense popularity. First, it is extremely easy to use since data can be easily entered, especially for algorithms that are adaptable to a table format. This is an important feature be-

cause it allows students to experiment with many numerical problems in a short period of time. Second, it includes high-level commands for two-dimensional and three-dimensional data visualization, and presentation graphics. Plots are easily obtained from within a script or in command mode. Third, the most evident power of a MATLAB is its speed of calculation. The program gives instantaneous feedback. Because of their popularity, MATLAB and other software such as MAPLE and Mathematica are now available in most university microcomputer laboratories.

One of the primary objectives of this text is to give students a clear step-by-step explanation of the algorithm corresponding to each numerical method used. To accomplish this objective, we have developed MATLAB m-functions contained in a supplementary CD-ROM at the back of the book. These M-files can be used for the application or illustration of all numerical methods discussed in this text. Each m-function will enable students to solve numerical problems simply by entering data and executing the m-functions.

It is well known that the best way to learn computer programming is to write computer programs. Therefore, I believe that by understanding the basic theory underlying each numerical method and the algorithm behind it, students will have the necessary tools to write their own programs in a high-level computer language such as C or FORTRAN.

1.2 AN INTRODUCTION TO MATLAB

In this section we give to the reader a brief tutorial introduction to MATLAB. For additional information we urge the reader to use the reference and user's guides of MATLAB.

1.2.1 Matrices and matrix computation

MATLAB treats all variables as matrices. They are assigned to expressions by using an equal sign and their names are case-sensitive. For example

```
>> A = [4 -2 5; 6 1 7; -1 0 6]
A =
```

```
   4   -2   5
   6    1   7
  -1    0   6
```

New rows may be indicated by a new line or by a semicolon. A column vector may be given as

```
>> x = [2; 8; 9] or x = [2  8  9]'
x =
     2
     8
     9
```

Elements of the matrix can be a number or an expression, like

```
>> x = [2    1+2    12/4    2^3]
x =
    2    3    3    8
```

One can define an array with a particular structure by using the command

```
x = a : step : b
```

As example

```
>> y = [0: 0.2 : 1]
y =
        0   0.2000   0.4000   0.6000   0.8000   1.0000

>> y = [0: pi/3 : pi]
y =
        0   1.0472   2.0944   3.1416

>> y = [20: -5 : 2]
y =
          20   15   10   5
```

MATLAB has a number of special matrices that can be generated by built-in functions

```
>> ones(2)
```

```
ans =
        1     1
        1     1
```

a matrix of all 1's.

```
>> zeros(2,4)
ans =
        0   0   0   0
        0   0   0   0
```

a 2×4 matrix of zeros.

```
>> rand(2,4)
ans =
        0.9501   0.6068   0.8913   0.4565
        0.2311   0.4860   0.7621   0.0185
```

a 2×4 random matrix with uniformly distributed random elements.

```
>> eyes(3)
ans =
        1   0   0
        0   1   0
        0   0   1
```

the 3×3 identity matrix.

The most useful matrix functions are

eig	eigenvalues and eigenvectors
inv	inverse
lu	LU decomposition
qr	QR factorization
rank	rank

A component-by-component addition of two vectors of the same dimensions is indicated as

```
>> x = [1 5 8];
>> y = [2 4 3];
>> x + y
ans =
      3     9     11
```

The component-by-component multiplication of the vectors x and y is indicated as

```
>> x. * y
ans =
        2      20      24
```

The transpose of a matrix or a vector is denoted by a prime symbol. For example

```
>> x'
ans =
        1
        5
        8
```

The matrix operations of multiplication, power, and division are indicated as

```
>> B = [1 3; 4 2];

>> A = [2 5; 0 6];

>> A*B

ans =
        22      16
        24      12

>> A^2
ans =
        4       40
        0       36

>> A/B
ans =
        1.6000        0.1000
        2.4000       -0.6000

>> A.*B
ans =
        2       15
        0       12
```

Note that the three operations of multiplication, division, and power can operate elementwise by preceding them with a period.

1.2.2 Output format

While all computations in MATLAB are performed in double precision, the format of the displayed output can be controlled by the following commands:

format short	fixed point with 4 decimal places (the default)
format long	fixed point with 14 decimal places
format short e	scientific notation with 4 decimal places
format long e	scientific notation with 15 decimal places
format rat	approximation by ratio of small integers

Once invoked, the chosen format remains in effect until changed.

It is possible to make the printed output from a MATLAB function look good by using the disp and fprintf functions.

disp(X) displays the array X, without printing the array name. If X is a string, the text is displayed. For example

```
disp('The input matrix A is')
```

will display the text placed in single quotes. The function disp also allows us to print numerical values placed in text strings. For example

```
disp(['Newton's method converges after ',num2str(iter),' iterations'])
```

will write on the screen

```
Newton's method converges after 9 iterations.
```

Here the function num2str(iter) converts a number to a string.

The other function fprintf is more flexible and has the form

```
fprintf('filename','format',list)
```

where,

filename is optional; if it is omitted, output goes to the screen.

format is a format control string containing conversion specifications or any optional text. The conversion specifications control the output of array elements.

Common conversion specifications include:

%P.Qe	for exponential
%P.Qf	fixed point
%P.Qg	to automatically select the shorter of %P.Qe or %P.Qf

where P and Q are integers that set the field width and the number of decimal places, respectively.

list is a list of variable names separated by commas, and the special format \n, produces a new line.

For example, the statements

```
>> x = pi ; y = 4.679 ; z = 9 ;
>> fprintf('\n x = %8.7f\n y = %4.3f\n z = %1.0f\n',x,y,z)
```

prints on the screen

```
x = 3.1415927
y = 4.679
z = 9
```

1.2.3 Planar plots

Plotting functions is very easy in MATLAB. Graphs of two-dimensional functions can be generated by using the MATLAB command plot. You can, for example, plot the function $y = x^2 + 1$ over the interval $[-1, 2]$ with the following commands:

```
>> x = [-1: 0.02: 2];
>> y = x.^2 +1;
>> plot(x,y)
```

The plot is shown in Figure 1.1. To plot a function $f1$ defined by an M-file (see Subsection 1.2.6) in an interval $[a, b]$, we use the command

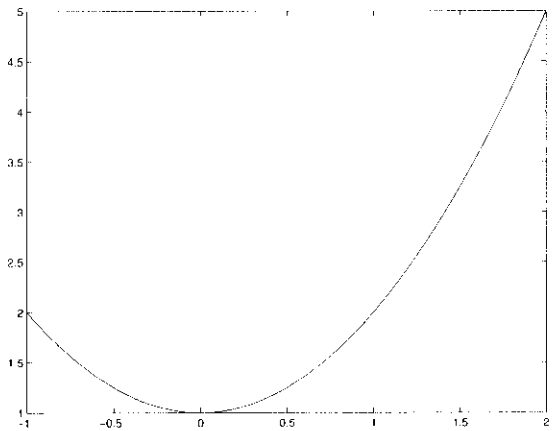


FIGURE 1.1
Plot of the function $f(x) = x^2 + 1$ in $[-1, 2]$.

```
>> fplot('f1',[a,b]).
```

Other features can be added to a given graph using commands such as grid, xlabel, ylabel, title, etc... For example, the command grid on will create grid lines on the graph.

Plots of parametrically defined curves can also be made. Try, for example,

```
t=[0: 0.01: 4*pi];  x=cos(2*t);  y=sin(3*t);  plot(x,y).
```

Multiple plots on a single graph are illustrated by

```
>> x=[0:0.1:2*pi];
>> y1=cos(x); y2=sin(x); y3=cos(2*x);
>> plot(x,y1,x,y2,x,y3)
```

1.2.4 3-D mesh plots

Three-dimensional or 3-D mesh surface plots are drawn with the function mesh. The command mesh(z) creates a three-dimensional perspective plot of the elements of the matrix z.

The following example will generate the plot shown in Figure 1.2, of the function $z = e^{-x^2-y^2}$ over the square $[-4,4] \times [-4,4]$.

```
>> [x y]=meshgrid(-4.0:0.2:4.0,-4.0:0.2:4.0);
>> z=exp(-x.^2-y.^2);
>> mesh(x,y,z)
```

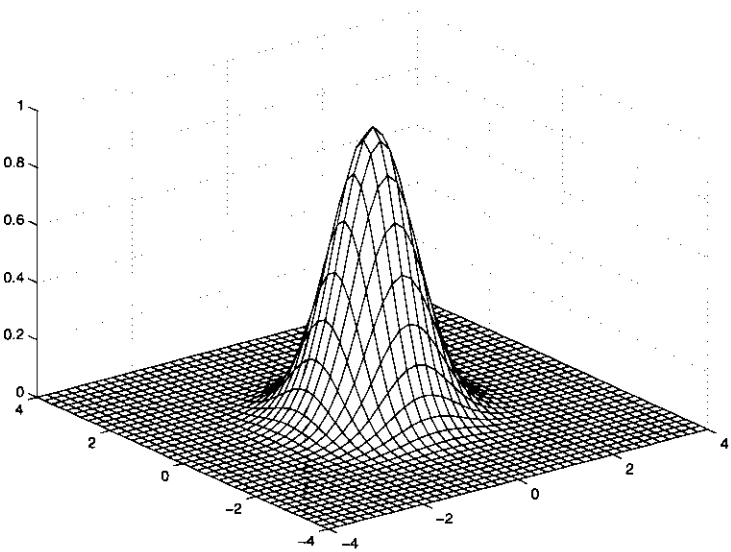


FIGURE 1.2
Three-dimensional surface.

Other commands related to 3-D graphics are plot3 and surf.

1.2.5 Function files

MATLAB contains a vast collection of built-in functions from elementary functions like sine and cosine, to more sophisticated functions like matrix inversions, matrix eigenvalues, and fast Fourier transforms. Table 1.1 contains a small selection of elementary mathematical MATLAB functions and Appendix B contains additional lists of functions grouped by subject area.

Note that all these function names must be in lower cases. MATLAB functions may have single or multiple arguments. For example,

cos(x)	cosine of x
sin(x)	sine of x
tan(x)	tangent of x
sqrt(x)	square root of x
abs(x)	absolute value of x
exp(x)	exponential of x
log(x)	log to the base e of x
log10(x)	log to the base 10 of x
cosh(x)	hyperbolic cosine of x
tanh(x)	hyperbolic tangent of x
asin(x)	inverse sine function of x
expm(A)	exponential of matrix A

Table 1.1 Elementary mathematical MATLAB functions.

```
[U,D]=eig(A)
```

produces a matrix U whose columns are the eigenvectors of A and a diagonal matrix D with the eigenvalues of A on its diagonal.

1.2.6 Defining functions

MATLAB allows users to define their own functions by constructing an M-file in the M-file Editor/Debugger. The first line of a function has the form

```
function y = function_name( input arguments)
```

As an example, let us define the function $f1(x) = 2x^3 - 3x + 1$ as an m.file. Using the Editor/Debugger we enter

```
function y=f1(x)
y=2*x.^3-3*x+1;
```

Once this function is saved as an M-file named f1.m, we can use the MATLAB command window to compute, for example,

```
>> f1(2)
ans =
        11
```

User defined functions can themselves have functions as input parameters. They can be evaluated internally by using the function feval. For example,

```
>> x1=fzero('f1',0)
x1=
        0.3949
```

gives the zero of f1 closest to 0.

1.2.7 Relations and loops

Like most computer languages, MATLAB offers a variety of flow control statements like for, while, and if. The statements that we use to control the flow are called relations.

The *Relational operators* in MATLAB are

==	equal
<=	less than or equal
>=	greater than or equal
~ =	not equal
<	less than
>	greater than

Note that "=" is used in an assignment statement while "==" is used in a relation.

Logical operators

&	and
|	or
~	not

if, for, while statements

The general form of an if loop is

```
if <if _expression>
      <statements>
elseif <if _expression>
      <statements>
            ...
else
      <statements>
end
```

Here is a simple example. If $i, j = 1, ..., 10$, then

```
if i==j
    A(i,j)=1;
else
    A(i,j)=0;
end
```

will result in a diagonal matrix A with 1's in its diagonal.

The general form of a for loop is

```
for <loop_variable= loop_expression>
      <statements>
end
```

For example,

```
S=0;
for i=1:100
    S=S+1;
end
S
```

results in the output

```
S =
    100
```

You can also nest for statements. The following statements create an $m \times n$ Hilbert matrix, H.

```
for i=1:m
    for j=1:n
        H(i,j)=1/(i+j-1);
    end
end
```

Finally, MATLAB also has its own version of the while loop, which repeatedly executes a group of statements while a condition remains true. Its form is

```
while < while _expression >
    < condition statements>
end
```

Given a positive number n the following statements compute and display the even powers of 3 less than n.

```
k=0;
while 3^k < n
    if rem(k,2)==0
        3^k
        k=k+1;
    else
        k=k+1;
    end
end
```

1.3 TAYLOR SERIES

THEOREM 1.1 (Taylor)
Suppose f has $n+1$ continuous derivatives on $[a,b]$. Let $x, x_0 \in [a,b]$. Then

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k + R_{n+1}(x) \tag{1.1}$$

where

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_0)^{n+1} \tag{1.2}$$

for some $\xi = \xi(x)$ *between* x_0 *and* x.

Eqn. (1.1) is called the **Taylor series** expansion of $f(x)$ about x_0 and $R_{n+1}(x)$ is called the **remainder** or the **truncation error**. Since in practical computation with Taylor series only a finite number of terms can be carried out, the term truncation error generally refers to the error involved in using a finite summation to approximate the sum of an infinite series.

EXAMPLE 1.1
Derive the Taylor series for $f(x) = e^x$ near $x_0 = 0$.

Since $f^{(k)}(0) = e^0 = 1$ for $k = 0, 1, \ldots$, then from Eqn. (1.1) we have

$$\begin{aligned} e^x &= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \frac{e^\xi}{(n+1)!}x^{n+1} \\ &= \sum_{k=0}^{n} \frac{x^k}{k!} + \frac{e^\xi}{(n+1)!}x^{n+1}. \end{aligned} \tag{1.3}$$

EXAMPLE 1.2
Using Example 1.1, approximate e with $n = 3$.

Set $x = 1$ and $n = 3$ in (1.3) to get

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} = 2.66667.$$

Now using (1.2), one can get an estimate of the truncation error. We have the result

$$R_4 = \frac{e^\xi}{4!} \quad \text{with} \quad 0 < \xi < 1$$

Since $\left|e^{\xi}\right| \leq e \leq 2.8$, the error is no greater in magnitude than

$$(1/4!)(2.8) = 0.11667.$$

If we let $x_0 = 0$ in Eqn. (1.1), we obtain the series known as **Maclaurin series**

$$f(x) = \sum_{k=0}^{n} \frac{f^k(0)}{k!} x^k + \frac{f^{(n+1)}(\xi)}{(n+1)!} x^{n+1} \tag{1.4}$$

with $\xi(x)$ between 0 and x.

EXAMPLE 1.3

Derive the Maclaurin series for $f(x) = \sin x$.

The derivatives of $\sin x$ are

$$\begin{aligned} f'(x) &= \cos x, & f'(0) &= 1, \\ f''(x) &= -\sin x, & f''(0) &= 0, \\ f'''(x) &= -\cos x, & f'''(0) &= -1, \\ \cdots &= \cdots & \cdots &= \cdots \end{aligned}$$

Therefore,

$$\sin x = \sum_{k=0}^{n} (-1)^k \frac{x^{2k+1}}{(2k+1)!} + (-1)^n \frac{x^{2n+3}}{(2n+3)!} \cos(\xi).$$

Another useful form of the Taylor series is obtained by replacing in Eqn. (1.1) x by $x + h$ and x_0 by x. The result is

$$f(x+h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + \frac{f^{(n+1)}(\xi)}{(n+1)!} h^{n+1} \tag{1.5}$$

with $\xi(h)$ between x and $x + h$.

Here we introduce a commonly used notation for the error term. We say that $F(h)$ *is of the order of* h^n and write

$$F(h) = O(h^n)$$

if there exists a real constant $C > 0$ and a positive integer n such that

$$|F(h)| \leq C\,|h^n| \qquad \text{for sufficiently small } h.$$

EXAMPLE 1.4
In Example 1.3 we found that for $n = 2$.

$$\sin x = x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 - \frac{1}{7!}x^7 \cos(\xi), \qquad 0 < \xi < x.$$

Since

$$\left| \frac{\sin x - x + \frac{1}{3!}x^3 - \frac{1}{5!}x^5}{x^7} \right| = \left| \frac{1}{7!} \cos(\xi) \right| \leq \frac{1}{7!} = C$$

then

$$\sin x - x + \frac{1}{3!}x^3 - \frac{1}{5!}x^5 = O(x^7).$$

The following theorem is a special case of Taylor's Theorem ($n = 0$).

THEOREM 1.2 (Mean Value Theorem)
If f is continuous on $[a, b]$ and is differentiable on (a, b), then there exists a number $c \in (a, b)$ such that (see Figure 1.3)

$$f(b) - f(a) = f'(c)(b - a).$$

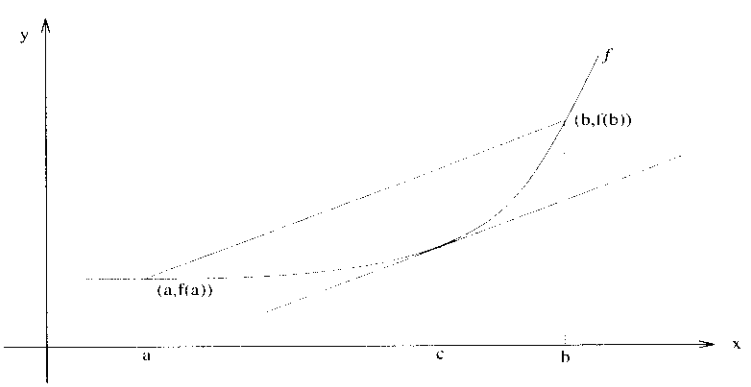


FIGURE 1.3
Mean Value Theorem.

The next theorem can be found in any calculus text and is an important tool for determining the existence of the solution of certain problems.

THEOREM 1.3 (Intermediate Value Theorem)
Suppose f is continuous on a closed interval $[a, b]$. If w is any number between $f(a)$ and $f(b)$, then there exists at least one c in (a, b) such that $f(c) = w$ (See Figure 1.4).

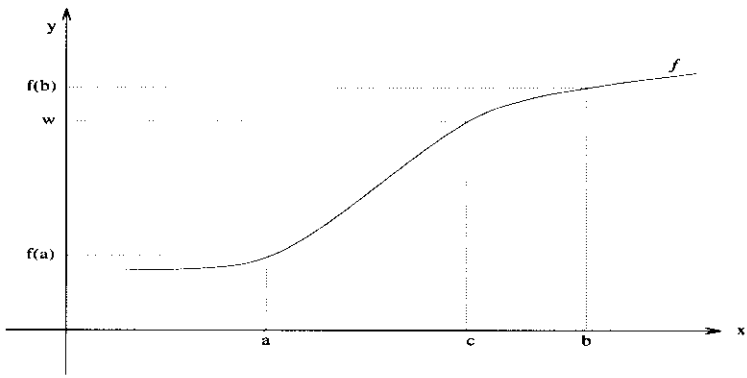


FIGURE 1.4
Intermediate Value Theorem.

EXAMPLE 1.5

Show that the equation $x^5 - 5x^4 - 7x^3 - x + 1 = 0$ has a solution in the interval $[0, 1]$.

Consider the function $f(x) = x^5 - 5x^4 - 7x^3 - x + 1$. It is easy to see that f is continuous on $[0, 1]$ and $f(0) = 1, f(1) = -11$. Since $f(1) < 0 < f(0)$, then, from the intermediate value theorem (IVT), there exists x in $(0, 1)$ such that

$$x^5 + 5x^4 - 7x^3 - x + 1 = 0.$$

EXERCISE SET 1.3

1. Find the Taylor series expansion of $f(x) = e^{-x}$ about $x_0 = 0$ and approximate $f(-0.88)$ with $n = 3$.

2. Find the Taylor series expansion for $f(x) = \ln(1 - x)$ about $x_0 = 0$.

3. Approximate $f(x) = e^x \sin x$ at $x = 0.2$ by using the Taylor series.

4. Find the Taylor series expansion of $\cos x$ about $x = \pi/3$.

5. Show that $f(x) = 2e^{2x} + 4x - 3$ has at least one zero in $[0, 1]$.

6. Find the Maclaurin series for $f(x) = x^2 e^x$.

7. Determine whether the following functions satisfy the hypotheses of the Mean Value Theorem. If so, find a number c satisfying the conclusion of the Mean Value Theorem.

   (a) $f(x) = x^{1/3}$, on $[-1, 2]$.

   (b) $f(x) = x^{1/3}$, on $[0, 2]$.

   (c) $f(x) = |x|$, on $[-2, 2]$.

8. Find all numbers c in $[1, 3]$ that satisfy the conclusion of the Mean Value Theorem for
$$f(x) = 2x + \frac{3}{x}.$$

9. Does the Mean Value Theorem tell us anything about the function
$$f(x) = \left|x^2 - 1\right| \quad \text{in } [0, 2].$$

10. Consider the so-called cosine integral
$$Cin(x) = \int_0^x \frac{1 - \cos t}{t} dt.$$

    (a) Find the degree $2n$ Taylor polynomial with remainder $R(x)$ about $a = 0$. Give an estimate for $|R(x)|$ which still depends on x.

    (b) Write a subroutine $CIN(X, TOL, CA, N)$ with input parameters X, TOL computing an approximate value CA for $Cin(X)$ with error less than TOL. The program should use Taylor polynomial of degree N to compute CA. N should be

determined by the program as small as possible, depending on X and TOL (use the results of part (a)).
Test your program for $X = 1$ and $TOL = 0.25, 0.06, 10^{-4}$. The exact value is $Cin(1) = 0.239811742$. Print in every case the error $CA - Cin(X)$, and N. Check if the actual errors are really $\leq TOL$. Print your results CA and the value of N for $X = 0.5$, $TOL = 10^{-4}$.

Chapter 2

Number System and Errors

The way in which numbers are stored and manipulated when arithmetic operations are performed on microcomputers is different from the way we, humans, do our arithmetic. We use the so-called decimal number system, while in microcomputers, internal calculations are done in the binary system. In this chapter we consider methods for representing numbers on computers and the errors involved.

2.1 FLOATING-POINT ARITHMETIC

In order to understand the major sources of errors involved in numerical solutions, we need to examine the manner in which numbers are stored in the computer. Since only numbers with a finite number of digits can be represented in the computer, floating-point form is used for the representation of real numbers.

A k-floating-point number in base β has the form

$$x = \pm\,(.b_1 b_2 \ldots b_k) \times \beta^e \tag{2.1}$$

where $\beta, b_1, \ldots, b_k$ are integers such that $\beta \geq 2$ and $0 \leq b_i \leq \beta - 1$. Here $(b_1 b_2 \ldots b_k)$ is called the mantissa and e the exponent. If $b_1 \neq 0$, or else $b_1 = b_2 = \cdots = b_k = 0$ such a floating-point number is said to be normalized.

β varies from computer to computer, but since most computers use the binary system ($\beta = 2$), we consider the normalized binary form

$$x = \pm\,(.1 b_2 \ldots b_k) \times 2^e \tag{2.2}$$

where the mantissa is expressed as a sequence of zeros and ones.

FIGURE 2.1
Internal representation of a 32-bit word length microcomputer.

As an abbreviation, the two binary digits 0 and 1 are usually called bits and the fixed-length group of binary bits is generally called a computer word. As an example, let us consider the floating-point number system of a 32-bit word length microcomputer. The internal representation of a word is shown in Figure 2.1. The leftmost bit is used for the sign of the mantissa, where the zero bit corresponds to a plus sign, and the unit bit to a negative sign. The next seven bits are used to represent the exponent with the first bit used as a sign to indicate whether the exponent is positive or negative. The final 24 bits represent the normalized mantissa with $b_1 = 1$.

EXAMPLE 2.1
Consider the machine number.

1	0000010	11010000000000000000000

The number is negative since the leftmost bit is 1. The exponent is positive and is equal to

$$0 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 0 \times 2^3 + 0 \times 2^4 + 0 \times 2^5 = (2)_{10}.$$

The final 24 bits indicate that the normalized mantissa is

$$1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-4} = (0.8125)_{10}.$$

Thus, this machine number represents the decimal number

$$-.8125 \times 2^2 = (-3.25)_{10}.$$

EXAMPLE 2.2

Write the decimal number 42.78125 in the normalized binary form.

We have

$$(42.)_{10} = (101010.)_2 \quad \text{and} \quad (.78125)_{10} = (.110010)_2.$$

Now

$$(42.78125)_{10} = (101010.110010)_2 = (0.101010110010)_2 \times 2^6.$$

Since the exponent $(6)_{10} = (110)_2$, the internal representation of 42.78125 in normalized binary form is

$$(00000110101010110010000000000000)_2.$$

The largest normalized positive machine number is

$$0 \quad 0111111 \quad 111111111111111111111111 \quad \approx 2^{63} \approx 10^{18}.$$

Thus, the range of numbers that can be stored is from about 10^{-19} to almost 10^{18} in decimal magnitude.

Numbers occurring in calculations that have a magnitude outside the above computer range result in what is called an **underflow** or an **overflow**.

The internal representation of a number that we have just described is an example of how numbers can be stored in a 32-bit word length microcomputer. However, a widely used internal representation of numbers in almost all new computers is the **IEEE standard for binary**

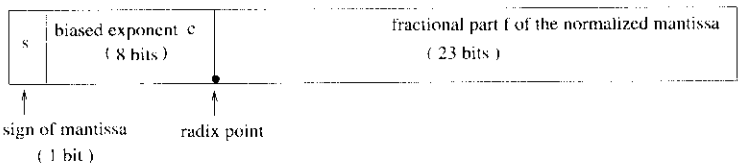


FIGURE 2.2
IEEE 32-bit format.

floating-point arithmetic. The single-precision format uses 32 bits and operates as follows:

$$\text{Floating-point number} = (-1)^s \times (1.f)_2 \times (2^{c-127})_{10}$$

The first bit is reserved for the sign bit s where $s = 0$ corresponds to $+$ and $s = 1$ corresponds to $-$. The next 8 bits are reserved for the **biased** or **offset** exponent c. The value of c ranges from $(0000\,0000)_2 = (0)_{10}$ to $(1111\,1111)_2 = (255)_{10}$. Finally, the remaining 23 bits are used for the mantissa. Note that only the part of the mantissa denoted by f is stored because the first bit in the mantissa is always equal to 1 so that this bit does not need to be stored. The IEEE 32-bit format is illustrated in Figure 2.2.

EXAMPLE 2.3
Determine the IEEE format of the decimal number -45.8125.

We have

$$(-45.)_{10} = (-101101)_2 \quad \text{and} \quad (.8125)_{10} = (110100)_2.$$

Hence,

$$(-45.8125)_{10} = (-101101.110100)_2 = (-1.011011101)_2 \times 2^5$$

The value of c is $c = 127 + 5 = (132)_{10}$ whose 8-bit form is $(10000100)_2$ and $s = 1$ since the number is negative. Thus, the IEEE format of -45.8125 is

1	1000 0100	0110 1110 1000 0000 0000 000

The largest number in the IEEE 32-bit format is obtained when the mantissa $(1.f)$ has a 1 in each bit position and the biased exponent $c = (1111\,1110)_2 = (254)_{10}$. The value of this number is therefore $(2 - 2^{-23})2^{127} \approx 3 \times 10^{38}$. The smallest positive number is $2^{-126} \approx 10^{-38}$.

The IEEE **double-precision** format employs 64 bits, of which 11 bits are reserved for the biased exponent, and 52 bits for the fractional part f of the normalized mantissa. The leading bit remains the sign bit.

EXERCISE SET 2.1

1. Write the following decimal numbers in the normalized binary form:
   (a) $(3.4375)_{10}$ (b) $(12.875)_{10}$

2. Determine the representation of the following decimal numbers in the floating-point number system of a 32-bit word length microcomputer.
   (a) $(5.25)_{10}$ (b) $(-3.84375)_{10}$

3. Convert the following 32-bit machine numbers to decimal

   (a) $a =$

1	0001010	11010100000000000000000

   (b) $b =$

0	0000110	10001010000000000000000

   (c) $c =$

1	0001110	11011000000000000000000

4. On a computer, floating-point numbers are represented in the following way

t	e	u	m

t = sign of exponent, 1 bit
e = absolute value of exponent, 3 bits
u = sign of mantissa, 1 bit
m = mantissa, 5 bits
The base is 2 and the mantissa is normalized.

(a) Give the largest number that can be represented exactly.

(b) Determine the decimal number corresponding to the word

$$(0101010110)_2.$$

5. Determine the IEEE format of the following decimal numbers

(a) 351.78215, (b) −0.5625.

2.2 ROUND-OFF ERRORS

There are two commonly used ways of representing a given real number x by a floating-point machine number, denoted by $fl(x)$, **rounding** and **chopping**. Consider a positive real number x in the normalized decimal form

$$x = (0.b_1 b_2 \ldots b_k b_{k+1} \ldots)_{10} \times 10^c \tag{2.3}$$

We say that the number x is chopped to k digits when all digits following the kth digits are discarded; that is, the digits $b_{k+1} b_{k+2} \ldots$ are chopped off to obtain

$$fl(x) = (0.b_1 b_2 \ldots b_k)_{10} \times 10^c.$$

Conversely, x is rounded to k digits when $fl(x)$ is obtained by choosing $fl(x)$ nearest to x; that is, adding one to b_k if $b_{k+1} \geq 5$ and chop off all but the first k digits if $b_{k+1} < 5$.

EXAMPLE 2.4
Consider the number e in the decimal system

$$e = 0.2718281828\ldots \times 10^1.$$

If $k = 7$ and rounding is employed, then

$$fl(e) = (0.2718281 + 0.0000001) \times 10^1 = 2.718282.$$

If, on the other hand, chopping is employed, then

$$fl(e) = 0.2718281 \times 10^1 = 2.718281.$$

The error that results from replacing a number x with its floating-point form $fl(x)$ is called **round-off error**. It is equal to the difference between x and $fl(x)$. The round-off error depends on x and is therefore best measured relative to the size of x.

DEFINITION 2.1 *If $x \neq 0$, the* **absolute error** *is given by $|x - fl(x)|$ and the* **relative error** *is given by $|x - fl(x)| \,/\, |x|$.*

As an example, let us find the relative error for the machine representation of x, given by Eqn. (2.3), when chopping is used to k digits. We have

$$\left|\frac{x - fl(x)}{x}\right| = \left|\frac{0.b_1b_2\cdots b_k\cdots \times 10^n - 0.b_1b_2\cdots b_k \times 10^n}{0.b_1b_2\cdots b_k\cdots \times 10^n}\right|$$

$$= \left|\frac{0.b_{k+1}b_{k+2}\cdots}{0.b_1b_2\cdots}\right| \times 10^{-k} \leq \frac{1}{0.1} \times 10^{-k}.$$

Thus,

$$\left|\frac{x - fl(x)}{x}\right| \leq 10^{-k+1}.$$

In a similar manner when rounding is used, the relative error is

$$\left|\frac{x - fl(x)}{x}\right| \leq 0.5 \times 10^{-k+1}.$$

For a binary computer with k bits in the mantissa, the bound on the relative error is 2^{-k} for rounding and 2^{1-k} for chopping.

There are situations when the round-off errors are so large that the accuracy of the numerical solution becomes questionable. An example of such a situation is the evaluation of polynomials of high degree near their zeros. Just having 24 bits for the mantissa in this case is not enough. One way of improving the accuracy of the solution is to use **double precision arithmetic**. This is done by adding another computer word to the mantissa. The result will provide the mantissa with 56 bits, that is 17 decimal digits of accuracy. A disadvantage in using double precision is that it increases the memory size required by the computer program and it consumes more computer time.

To see how round-off can drastically reduce the accuracy in some cases, let us consider the two numbers:

$$x = 1.23456 \qquad \text{and} \qquad y = 1.22222.$$

These two numbers are nearly equal and both carry six decimal digits of precision. If x and y are replaced by their first three digits, then

$$x - y = 1.23 - 1.22 = 0.01.$$

Thus, the error is about 1% in each. But if we compute the difference

$$1.23456 - 1.22222 = 0.01234$$

we can see that 0.01 differs from 0.01234 by about 20%. Similarly, when numbers are accurate to one part in 10^8, subtraction of nearly equal numbers can result in a large percent error compared with their actual difference.

We end this section with the following definition:

DEFINITION 2.2 *If x^* is an approximation to x, then we say that x^* approximates x to k* **significant digits** *if k is the largest non-negative integer for which*

$$\left|\frac{x - x^*}{x}\right| < 5 \times 10^{-k}.$$

Take for example, $x = 3.1415$. For x^* to approximate x to three significant digits, it has to satisfy

$$\left|\frac{3.1415 - x^*}{3.1415}\right| < 5 \times 10^{-3}.$$

That is,

$$3.13 < x^* < 3.15.$$

EXERCISE SET 2.2

1. Perform the following operations using
   (i) four-digits rounding arithmetic
   (ii) four-digits chopping arithmetic
   (a) $\frac{1}{3} + \frac{7}{4}$ (b) $\left(\frac{2}{3} - \frac{4}{7}\right) + \left(\frac{8}{5} - \frac{13}{7}\right)$ (c) $\left(\frac{5}{3} \times \frac{2}{7}\right) + \frac{2}{3}$

2. Let $x = 0.86439868$ and $y = 0.86433221$. Compute the relative error for the following operations if the operations were done using five significant digits and rounding:
   (a) $fl(x) + fl(y)$ (b) $fl(x) \times fl(y)$

3. If x_1, x_2 are approximations to X_1, X_2 with errors ϵ_1, ϵ_2, show that the relative error

$$\frac{X_1X_2 - x_1x_2}{X_1X_2}$$

   of the product X_1X_2 is approximately the sum $\frac{\epsilon_1}{X_1} + \frac{\epsilon_2}{X_2}$.

4. Find the roots of the quadratic equation $x^2 + 6x + 2 = 0$, using four digits and rounding.

5. Given the numbers

$$a = 0.67323, \quad b = 12.751, \quad \text{and} \quad c = 12.687,$$

   write the quotient $\frac{a}{b-c}$ in the form $\frac{a(b+c)}{b^2-c^2}$ and compute it using the numbers as given, and then using three digits and rounding.

2.3 TRUNCATION ERROR

A round-off error arises in terminating the decimal expansion and rounding. In contrast, the truncation error terminates a process. So,

the **truncation error** generally refers to the error involved in terminating an infinite sequence or series after a finite number of terms, that is, replacing non-computable problems with computable ones. To make the idea precise, consider the familiar infinite Taylor series (Section 1.3)

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

which can be used to approximate the cosine of an angle x in radians.

Let us find, for instance, the value of $\cos(0.5)$ using three terms of the series. We have

$$\cos(0.5) \approx 1 - \frac{1}{8} + \frac{1}{384} \approx 0.8776041$$

thus,

$$\text{truncation error} = \cos(0.5) - .8776041 \approx .216 \times 10^{-4}$$

Note that one can use the expression of the remainder R_{n+1} of the Taylor series to get a bound on the truncation error.

EXERCISE SET 2.3

1. Find the Maclaurin series for $f(x) = e^x$. Use three terms of the series to find an approximation to $e^{0.5}$ and the truncation error.

2. Using the Taylor series expansion for $\cos x$, approximate $\cos(1.7)$ to an accuracy of 10^{-4}.

3. Approximate $\ln 0.5$ using three terms of the Taylor series expansion of $\ln(1 - x)$ about 0.

2.4 INTERVAL ARITHMETIC

Most practical problems requiring numerical computation involve quantities determined experimentally by approximate measurements which are often estimated to a certain number of significant decimal

digits. Results computed from such inexact initial data will also be of a limited precision. One way of dealing with such a problem is to replace the inexact initial data with intervals known in advance to contain the desired exact results. Our main objective is then to narrow such intervals. Interval arithmetic is also used to construct mathematically rigorous error bounds around approximate solutions. The property that interval evaluations bound the range is also useful in various contexts. In this section we will introduce and discuss an arithmetic for intervals.

To begin, we define an interval number to be the ordered pair of real numbers, $[a_1, a_2]$, with $a_1 < a_2$. The interval $[a_1, a_2]$ is also a set of real numbers x such that $a_1 \leq x \leq a_2$. We can therefore write

$$[a_1, a_2] = \{\, x \,|\, a_1 \leq x \leq a_2 \},$$

$x \in [a_1, a_2]$ means x is a real number in the interval $[a_1, a_2]$ and $[b_1, b_2] \subset [a_1, a_2]$ means that the interval $[b_1, b_2]$ is contained as a set in the interval $[a_1, a_2]$, that is, $a_1 \leq b_1 \leq b_2 \leq a_2$.

We now introduce some further notation which will be used in this section. We denote the set of all closed real intervals by $I(\Re)$ and the elements of $I(\Re)$ by $A, B, ..., Z$. Intervals of the form $[a_1, a_1]$ are equivalent to the real number a_1.

DEFINITION 2.3 *If $A = [a_1, a_2]$, $B = [b_1, b_2] \in I(\Re)$, then*

1.
$$A = B \quad \Longleftrightarrow \quad a_1 = b_1 \text{ and } a_2 = b_2.$$

2. *The absolute value or "magnitude" of A is defined as*
$$|A| = \max\{\, |a_1|,\, |a_2| \}.$$

3. *The width of A is defined to be*
$$d(A) = a_2 - a_1 \geq 0.$$

4. *A partial ordering of the elements of $I(\Re)$ is defined by*
$$A < B \quad \Longleftrightarrow \quad a_2 < b_1.$$

We now introduce arithmetic operations on elements from $I(\Re)$.

DEFINITION 2.4 *Let $*$ be one of the symbols $\{+, -, \cdot, \div\}$. If $A, B \in I(\Re)$, we define arithmetic operations on intervals by*

$$A * B = \{x * y \mid x \in A, y \in B\} \tag{2.4}$$

except that we do not define A/B if $0 \in B$.

The definition of the interval arithmetic operations given by Eqn. (2.4) emphasizes the fact that the sum, difference, product, or quotient of two intervals is the set of sums, differences, products, or quotients, respectively, of pairs of real numbers, one from each of the two intervals.

In general, the new set of possible values of x that result from the operation on intervals $A = [a_1, a_2]$ and $B = [b_1, b_2]$ may be found explicitly as

$$\begin{aligned} A + B &= [a_1 + b_1, a_2 + b_2], \\ A - B &= [a_1 - b_2, a_2 - b_1], \\ A \cdot B &= [\min\{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}, \max\{a_1b_1, a_1b_2, a_2b_1, a_2b_2\}], \\ A \div B &= [a_1, a_2] \cdot [1/b_2, 1/b_1] \text{ provided that } 0 \notin [b_1, b_2]. \end{aligned} \tag{2.5}$$

EXAMPLE 2.5

Let $A = [-1, 2]$ and $B = [1, 3]$. The operations mentioned above are now illustrated for these two intervals.

$$\begin{aligned} [-1,2] + [1,3] &= [0,5] \\ [-1,2] - [1,3] &= [-4,1] \\ [-1,2] \cdot [1,3] &= [\min\{-1,-3,2,6\}, \max\{-1,-3,2,6\}] \\ &= [-3,6] \\ [-1,2] \div [1,3] &= [-1,2] \cdot [1/3,1] = [-1,2]. \end{aligned}$$

From the definition given by 2.4 it is clear that interval arithmetic is an extension of real arithmetic. It follows directly from definition 2.4 that

the interval addition and interval multiplication are both commutative and associative. That is, if A, B and $C \in I(\Re)$, then it follows that

$$A + B = B + A, \quad A \cdot B = B \cdot A \qquad \text{(commutativity)}$$

$$(A+B)+C = A+(B+C), \quad (A \cdot B) \cdot C = A \cdot (B \cdot C) \qquad \text{(associativity)}$$

The real numbers 0 and 1 are identities for interval addition and multiplication, respectively. That is, for any $A \in I(\Re)$, we have

$$0 + A = A + 0 = A$$
$$1 \cdot A = A \cdot 1 = A$$

In general, the distributive law does not hold for interval arithmetic. In order to show that, consider the example

$$[1,3] \cdot ([1,4] - [1,4]) = [1,3] \cdot ([-3,3]) = [-9,9]$$

whereas

$$\begin{aligned}[1,3] \cdot ([1,4] - [1,4]) &\neq [1,3] \cdot [1,4] - [1,3] \cdot [1,4] \\ &\neq [1,12] - [1,12] = [-11,11].\end{aligned}$$

But the *subdistributivity* holds for any interval A, B, and C. That is,

$$A \cdot (B + C) \subset A \cdot B + A \cdot C, \qquad \text{subdistributivity.}$$

Proof:
$$\begin{aligned}A \cdot (B + C) &= \{w = x\,(y + z) \,|\, x \in A,\, y \in B,\, z \in C\} \\ &\subseteq \{v = x\,y + \bar{x}\,z \,|\, x,\, \bar{x} \in A,\, y \in B,\, z \in C\} \\ &= AB + AC.\end{aligned}$$

We now want to turn to an application of the interval arithmetic for computing the range of real functions. We have the following results:

If $f(x_1, x_2, ..., x_n)$ is a real rational expression in which each variable x_i appears only once and to the first power, than the corresponding interval expression $F(X_1, X_2, ..., X_n)$, *called interval evaluation for* f, will compute the actual range of values of f for $x_i \in X_i$ (see [2]). Here we assume that all occurring expressions are composed of finitely many operations for which the interval arithmetic operations in F are defined.

EXAMPLE 2.6
Consider the two expressions for the function $f(x, a) = 1 - a/x$, with $x \in [2, 3]$

$$f_1(x, a) = 1 - \frac{a}{x}$$
$$f_2(x, a) = \frac{x - a}{x}$$

Using interval arithmetic, the interval evaluations for f_1 and f_2 when $a \in [0, 1]$ are

$$F_1([2, 3], [0, 1]) = 1 - \frac{[0, 1]}{[2, 3]} = \left[\frac{1}{2}, 1\right],$$

$$F_2([2, 3], [0, 1]) = \frac{[2, 3] - [0, 1]}{[2, 3]} = \left[\frac{1}{3}, \frac{3}{2}\right]$$

The range of all values of f when $x \in [2, 3]$ and $a \in [0, 1]$ is $\left[\frac{1}{2}, 1\right]$. From the above results, it is clear that the interval evaluation of a function f is dependent on the choice of expression for f. In this example, the range obtained by the expression F_2 is different from the actual range of f because of the appearance of the variable x twice in the expression. For a continuous function of a real variable, we have the following result:

Let f be a continuous function of the real variable x. Furthermore, we assume that the interval evaluation $f(Y)$ is defined for the interval Y. It follows that

$$\text{for } X \subseteq Y,$$

and

$$\text{actual range of } f = \{f(x) | \, x \in X\} \subseteq f(X).$$

EXAMPLE 2.7
Consider the function

$$f(x) = \frac{x - 3}{x + 2}, \quad x \neq -2.$$

For $x = [2, 4]$, we have

$$\{f(x) | \, x \in [2, 4]\} = [-\frac{1}{4}, \frac{1}{6}] \subset f([2, 4]) = [-\frac{1}{4}, \frac{1}{4}].$$

Another important application of interval arithmetic is the solvability of the equation

$$A\,X = B \tag{2.6}$$

where $X \in I\,(\Re)$ and $A \neq [0, 0]$. The solution set of the equation is

$$\{\, x = b/a | \, a \in [a_1, a_2], \;\; b \in [b_1, b_2]\}.$$

EXAMPLE 2.8

Consider the equation

$$[1, 3]\,X = [-1, 4].$$

The solution set of the equations $ax = b$ with $a \in [1, 3]$ and $b \in [-1, 4]$ is given by

$$\{\, x = b/a | \, a \in [1, 3], \;\; b \in [-1, 4]\} = [-1, 4]/[1, 3] = [-1, 4]$$

which is different from the unique interval solution $X = [-\frac{1}{3}, \frac{4}{3}]$ of the equation $AX = B$. Note that $[-\frac{1}{3}, \frac{4}{3}] \subset [-1, 4]$. In general, one can show that

$$X \subseteq B/A$$

as follows:

if $z \in X$, then there exists $a \in A$ and $b \in B$ such that

$$az = b \Rightarrow \;\; z = b/a \in B/A.$$

The starting point for the application of interval analysis was, in retrospect, the desire in numerical mathematics to be able to execute algorithms on digital computers capturing all the round-off errors automatically and therefore to calculate strict error bounds automatically. Interval arithmetic, when practical, allows rigor in scientific computations and can provide tests of correctness of hardware, and function libraries for floating-point computations. A complete study of interval arithmetic can be found in [2]. In that reference, more topics such as interval matrix operations and interval systems of equations are presented.

EXERCISE SET 2.4

1. Carry out the following interval arithmetic:

   (a) $[0, 2] + [2, 3]$,

   (b) $[2.0, 2.2] - [1.0, 1.1]$,

   (c) $[-5, -3] \cdot [-7, 4]$,

   (d) $[-2, 1] \cdot [-4.4, 2.1]$,

   (e) $[-2, 2]/[-3/2, -1]$,

   (f) $3 \cdot [-2, 4]$,

   (g) $[-1, 0]/[1, 2] + [-2, 3] \cdot [4, 5]$.

2. Using interval arithmetic, compute bounds on the range of the following real rational functions:

   (a) $f(x) = x^2 + 2x + 1, \quad 0 \le x \le 1.$

   (b) $f(x, y, z) = xy + z, \quad -1 \le x \le 1, \ -1 \le y \le 1, \ 1 \le z \le 2.$

   (c) $f(x) = x/(1 + x), \quad 0 \le x \le 1.$

3. Compute a solution set for $[2, 4]X = [-2, 1]$.

4. By testing the sign of the end points of the intervals $A = [a, b]$ and $B = [c, d]$, break the formula for interval multiplication into nine cases only one of which requires more than two multiplications.

5. Using the results of the previous exercise carry out the following interval arithmetic:

   (a) $[1, 2] \cdot [3, 4]$,

   (b) $[-1, 2] \cdot [-4, -3]$.

Chapter 3

Roots of Equations

In this chapter we shall discuss one of the oldest approximation problems which consists of finding the roots of an equation. It is also one of the most commonly occurring problems in applied mathematics. The **root-finding problem** consists of the following: given a continuous function f, find the values of x that satisfy the equation

$$f(x) = 0. \tag{3.1}$$

The solutions of this equation are called the **zeros** of f or the **roots** of the equation. In general, Eqn. (3.1) is impossible to solve exactly. Therefore, one must rely on some numerical methods for an approximate solution. The methods we will discuss in this chapter are iterative and consist basically of two types: one in which the convergence is guaranteed and the other in which the convergence depends on the initial guess.

3.1 THE BISECTION METHOD

Let $f(x)$ be a given function, continuous on an interval $[a, b]$, such that

$$f(a)f(b) < 0. \tag{3.2}$$

Using the Intermediate Value Theorem, it follows that there exists at least one zero of f in (a, b). To simplify our discussion, we assume that f has exactly one root α. Such a function is shown in Figure 3.1.

The bisection method is based on halving the interval $[a, b]$ to determine a smaller and smaller interval within which α must lie. The procedure is carried out by first defining the midpoint of $[a, b]$, $c = (a + b)/2$

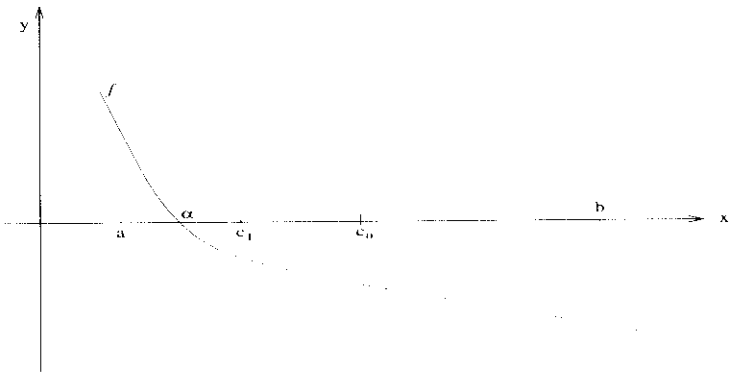


FIGURE 3.1
The bisection method and the first two approximations to its zero α.

and then computing the product $f(c)f(b)$. If the product is negative, then the root is in the interval $[c, b]$. If the product is positive, then the root is in the interval $[a, c]$. Thus, a new interval containing α is obtained. The process of halving the new interval continues until the root is located as accurately as desired, that is

$$|a_n - b_n| < \epsilon \tag{3.3}$$

where a_n and b_n are the endpoints of the nth interval $[a_n, b_n]$ and ϵ is a specified tolerance value.

Some other stopping criteria that one can use, other than (3.3), are given by

$$\frac{|a_n - b_n|}{|a_n|} < \epsilon \tag{3.4}$$

or

$$|f(a_n)| < \epsilon. \tag{3.5}$$

An algorithm statement of this method is shown below. Suppose $f(a)f(b) \leq 0$. Let $a_0 = a$ and $b_0 = b$.

for $n = 0, 1, \ldots,$ ITMAX
  $c \leftarrow \frac{a_n + b_n}{2}$
  if $f(a_n)f(c) \leq 0$, set $a_{n+1} = a_n$, $b_{n+1} = c$
  otherwise, set $a_{n+1} = c$, $b_{n+1} = b_n$

m-function 3.1

The following MATLAB function **bisect.m** finds the solution of an equation using the Bisection method. INPUTS are a function *f(x)*; the endpoints *a* and *b*; a tolerance *tol*; the maximum number of iterations *n*. The input function *f(x)* should be defined as an M-file.

```
function bisect(f,a,b,tol,n)
% Bisection method for solving the nonlinear equation f(x)=0.
a0=a; b0=b;
iter=0;
u=feval(f,a);
v=feval(f,b);
c=(a+b)*0.5;
err=abs(b-a)*0.5;
disp('_____________________________________________')
disp(' iter        a        b        c        f(c)      |b-a|/2  ')
disp('_____________________________________________')
fprintf('\n')
if (u*v<=0)
  while (err>tol)&(iter<=n)
    w=feval(f,c);
    fprintf('%2.0f  %10.4f  %10.4f  %12.6f  %10.6f  %10.6f\n', iter,
a, b, c, w, err)
    if (w*u<0)
      b=c;v=w;
    end;
    if (w*u>0)
      a=c;u=w;
    end;
    iter=iter+1;
    c=(a+b)*0.5;
    err=abs(b-a)*0.5;
  end;
  if (iter>n)
    disp('  Method failed to converge')
  end;
else
  disp('  The method cannot be applied f(a)f(b)>0')
end;
% Plot f(x) in the interval [a,b].
fplot(f, [a0 b0])
xlabel('x'); ylabel('f(x)'); grid
```

EXAMPLE 3.1
The function $f(x) = x^3 - x^2 - 1$ has exactly one zero in $[1, 2]$. Use the bisection algorithm to approximate the zero of f to within 10^{-4}.

Since $f(1) = -1 < 0$ and $f(2) = 3 > 0$, then (3.2) is satisfied. Starting with $a_0 = 1$ and $b_0 = 2$; we compute

$$c_0 = \frac{a_0 + b_0}{2} = \frac{1+2}{2} = 1.5 \quad \text{and} \quad f(c_0) = 0.125.$$

Since $f(1.5)f(2) > 0$, the function changes sign on $[a_0, c_0] = [1, 1.5]$. To continue, we set $a_1 = a_0$ and $b_1 = c_0$; so

$$c_1 = \frac{a_1 + b_1}{2} = \frac{1+1.5}{2} = 1.25 \quad \text{and} \quad f(c_1) = -0.609375.$$

Again $f(1.25)f(1.5) < 0$ so the function changes sign on $[c_1, b_1] = [1.25, 1.5]$. Next we set $a_2 = c_1$ and $b_2 = b_1$. Continuing in this manner leads to the values in Table 3.1 which converge to $r = 1.465454$.

Note that, before calling the MATLAB function bisect, we must define the MATLAB function f1 as follows:

```
function f=f1(x)
f=x.^3-x.^2-1;
```

Now call the the function bisect to get the results shown in Table 3.1.

Some of the features of MATLAB can be used to view, for example, step-by-step calculations involved in the function bisect. This can be done by using the pause function of the form pause(s). If we add the line

```
pause(1);
```

just before the end statement of the while loop, execution is held up for approximately 1 second at the end of each output line. MATLAB also provides a wide range of graphic facilities that may be used to graphically visualize the results. For example if the following commands

```
fplot('f1',[a0,b0])
grid
```

» bisect('f1',1,2,10^(-4),40)

iter	a	b	c	f(c)	|b-a|/2
0	1.0000	2.0000	1.500000	0.125000	0.500000
1	1.0000	1.5000	1.250000	-0.609375	0.250000
2	1.2500	1.5000	1.375000	-0.291016	0.125000
3	1.3750	1.5000	1.437500	-0.095947	0.062500
4	1.4375	1.5000	1.468750	0.011200	0.031250
5	1.4375	1.4688	1.453125	-0.043194	0.015625
6	1.4531	1.4688	1.460938	-0.016203	0.007813
7	1.4609	1.4688	1.464844	-0.002554	0.003906
8	1.4648	1.4688	1.466797	0.004310	0.001953
9	1.4648	1.4668	1.465820	0.000875	0.000977
10	1.4648	1.4658	1.465332	-0.000840	0.000488
11	1.4653	1.4658	1.465576	0.000017	0.000244
12	1.4653	1.4656	1.465454	-0.000411	0.000122

Table 3.1 Solution of $x^3 - x^2 - 1 = 0$ in $[1, 2]$ using the bisection method.

are added at the end of the function bisect.m, then a plot of the function $f(x) = x^3 - x^2 - 1$ in $[a, b]$, shown in Figure 3.2, will be given with the output.

Although the Bisection method always converges and is easy to program, compared to other methods, its rate of convergence is rather slow. We state the following convergence result.

THEOREM 3.1

Let f be a continuous function on an interval $[a, b]$ where $f(a)f(b) < 0$. At the nth step, the bisection procedure approximates a root with an error of at most $(b - a)/2^{n+1}$.

Proof: Let $[a_0, b_0]$ be the original interval and α a zero of f in this interval. Define

$$c_0 = \frac{b_0 + a_0}{2} \qquad \text{and} \qquad \alpha \in [a_0, b_0]$$

then

$$|\alpha - c_0| < (b_1 - a_1) = \frac{b_0 - a_0}{2}$$

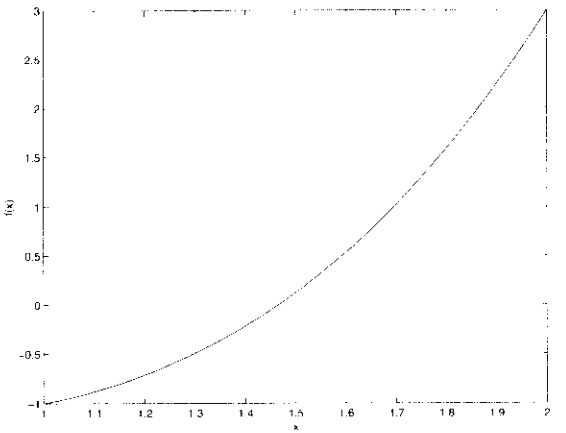


FIGURE 3.2
A plot of $f(x) = x^3 - x^2 - 1$ in $[1, 2]$.

with a_1 and b_1 being the endpoints of the new subinterval containing α.
If c_n denotes the nth value of c, then we have

$$|\alpha - c_n| < (b_{n+1} - a_{n+1}) = \frac{b_0 - a_0}{2^{n+1}}, \quad n = 0, 1, \ldots.$$

EXAMPLE 3.2

Determine the number of iterations required to find the zero of $f(x)$, defined in Example 3.1, with an absolute error of no more than 10^{-6}.

We want to find n such that

$$|\alpha - c_n| \leq \frac{b - a}{2^{n+1}} = \frac{2 - 1}{2^{n+1}} < 10^{-6}.$$

This implies that

$$-(n + 1)\log(2) < -6 \qquad \text{or} \qquad n > \frac{6}{\log 2} - 1 \approx 19.$$

So one needs at most 19 iterations to obtain an approximation accurate to 10^{-6}.

EXERCISE SET 3.1

1. Find a root of $f(x) = x^3 + 2x - 3$ in the range $0 \leq x \leq 7/5$ using the bisection method.

2. The function $f(x) = x^2 - 2.6x - 2.31$ has one root in the interval $[3, 4]$. How many steps of the bisection method are needed to locate the root with an error of at most 10^{-5}.

3. Find a root of the equation $e^{2x} - 7x = 0$.

4. Verify that the function $f(x) = x^2 \sin x + 2x - 3$ has exactly one root in $(0, 2)$. Find this root by using the bisection method with an error of no more than 10^{-5}. Count the number of iterations needed to find the root and compare your answer with the result obtained by using the error formula given by Theorem 1 of section 3.1.

5. The function $f(x) = x^5 - 3x^3 + 2.5x - 0.6$ has exactly one root in $(1, 2)$. Demonstrate it graphically and determine the root with an error of at most 10^{-5}.

6. Consider the function

$$f(x) = x^4 - 5x^3 + \frac{22}{3}x^2 - \frac{116}{27}x + \frac{8}{9}$$

   (a) Check that $f(x)$ has a root α_1 between 0 and 1 and another root α_2 between 1 and 4.

   (b) Compute both roots using the bisection method.

7. Sketch the graph of $f(x) = \tan x + \tanh x = 0$ and find an interval in which its smallest positive root lies . Determine the root correct to two decimal digits using the bisection method.

8. **Computer assignment:**
   Write a computer program in FORTRAN or C that finds the zeros of a function $f(x)$ on an interval $[a, b]$ using the bisection method. Assume that there is only one simple root in this interval. Input data to the program should be a function $f(x)$, a, b and the error tolerance ϵ.

What does your program do if the given f is $f(x) = \cos x$ and $a = 0, b = \pi$? What does it do if $a = 0, b = 2\pi$?

Test your program to find the root of the equation given in Exercise 2.

3.2 THE METHOD OF FALSE POSITION

The method of false position, also known as **regula falsi**, is similar to the Bisection method but has the advantage of being slightly faster than the latter. As in the bisection method, the function is assumed to be continuous over the interval $[x_1, x_2]$ with

$$f(x_1)f(x_2) < 0.$$

In this procedure, the first point x_3 is selected as point of intersection of the x-axis, and the straight line joining the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ as shown in Figure 3.3. From the equation of the secant line, it follows that

$$x_3 = x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}. \tag{3.6}$$

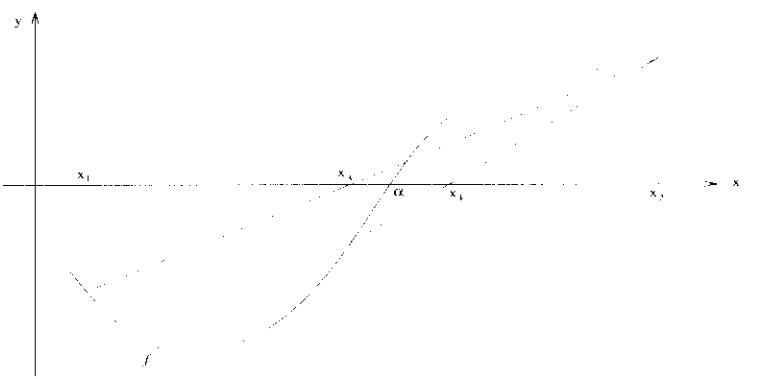


FIGURE 3.3
The method of false position and the first two approximations to its zero α.

With x_3 now known, we then compute $f(x_3)$ and repeat the procedure between the values at which the function changes sign, that is, if

$f(x_1)f(x_3) < 0$ set $x_2 = x_3$, otherwise set $x_1 = x_3$. At each step the method of false position produces a new interval that contains a root of f and the sequence of points generated by the method will eventually converge to the root. An algorithm of the method of *false position* is given below.

Given a continuous function $f(x)$ on $[a_0, b_0]$ with $f(a_0)f(b_0) < 0$.

for $n = 0, 1, \ldots, \text{ITMAX}$
$\quad c \leftarrow \frac{f(b_n)a_n - f(a_n)b_n}{f(b_n) - f(a_n)}$

$\quad$ **if** $f(a_n)f(c) < 0$, set $a_{n+1} = a_n$, $b_{n+1} = c$
$\quad$ **otherwise**, set $a_{n+1} = c$, $b_{n+1} = b_n$.

Because in some cases the method of false position fails to give a small interval in which the zero is known to lie, we terminate the iterations if

$$|f(x_n)| \leq \epsilon$$

or

$$|x_n - x_{n-1}| \leq \epsilon$$

where ϵ is a specified tolerance value.

One of the main disadvantages of this method is that if the sequence of points generated by its algorithm is one-sided, it will tremendously slow the convergence of the method. This applies to a function whose graph is concave up or concave down on $[a_0, b_0]$ (see Figure 3.4).

EXAMPLE 3.3

The function $f(x) = x^3 - x^2 - 1$ has exactly one zero in $[1, 2]$. Use the method of false position to approximate the zero of f to within 10^{-4}.

A root lies in the interval $[1, 2]$ since $f(1) = -1$ and $f(2) = 3$. Starting with $a_0 = 1$ and $b_0 = 2$, we get using Eqn. (3.6)

$$c_0 = 2 - \frac{3(2-1)}{3-(-1)} = 1.25 \quad \text{and} \quad f(c_0) = -0.609375.$$

Here, $f(c_0)$ has the same sign as $f(a_0)$ and so the root must lie on the interval $[c_0, b_0] = [1.25, 2]$. Next we set $a_1 = c_0$ and $b_1 = b_0$ to get the

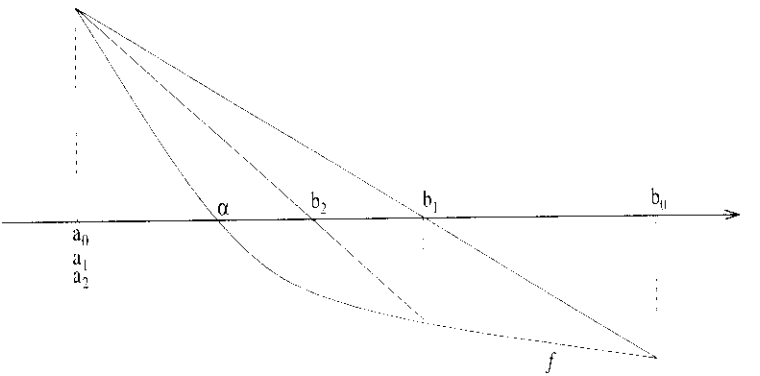


FIGURE 3.4
The method of false position and approximation lines on $[a_0, b_0]$.

next approximation

$$c_1 = 2 - \frac{3 - (2 - 1.25)}{3 - (-0.609375)} = 1.37662337 \quad \text{and} \quad f(c_1) = -0.2862640.$$

Now $f(x)$ changes sign on $[c_1, b_1] = [1.37662337, 2]$. Thus we set $a_2 = c_1$ and $b_2 = b_1$. Continuing in this manner leads to the values in Table 3.2 which converge to $r = 1.465558$.

EXERCISE SET 3.2

1. Find a root of $f(x) = x^3 + 2x - 3$ in the range $0 \le x \le 7/5$ using the method of false position.

2. Find a root of the equation $e^{2x} - 7x = 0$.

3. Verify that the function $f(x) = x^2 \sin x + 2x - 3$ has exactly one root in $(0, 2)$. Find this root using the method of false position with an error of no more than 10^{-3}.

4. The function $f(x) = x^5 - 3x^3 + 2.5x - 0.6$ has exactly one root in $(1, 2)$. Demonstrate it graphically and determine the root with an error of at most 10^{-4} using the method of false position.

m-function 3.2

The following MATLAB function **false.m** finds the solution of an equation using the method of false position. INPUTS are a function *f*; the endpoints *a* and *b*; a tolerance *tol*; the maximum number of iteration *n*. The input function *f(x)* should be defined as an M-file.

```
function false(f,a,b,tol,n)
% False position method for solving the nonlinear equation f(x)=0.
a0=a; b0=b;
iter=0;
u=feval(f,a);
v=feval(f,b);
c=(v*a-u*b)/(v-u);
w=feval(f,c);
disp('_____________________________________________')
disp(' iter        a        b        c        f(c)        |b-a|  ')
disp('_____________________________________________')
fprintf('\n')
if (u*v<=0)
  while (abs(w)>tol)&(abs(b-a)>tol)&(iter<=n)&((v-u)~=0)
    w=feval(f,c);
    fprintf('%2.0f %12.4f %12.4f  %12.6f  %10.6f %10.6f\n', iter, a,
b, c, w, abs(b-a))
    if (w*u<0)
      b=c;v=w;
    end;
    if (w*u>0)
      a=c;u=w;
    end;
    iter=iter+1;
    c=(v*a-u*b)/(v-u);
  end;
  if (iter>n)
    disp('  Method failed to converge')
  end;
  if (v-u==0)
    disp('  Division by zero')
  end;
else
  disp('  The method cannot be applied f(a)f(b)>0')
end;
fplot(f,[a0 b0])
xlabel('x');ylabel('f(x)'); grid
```

» false('f1',1,2,10^(-4),40)

iter	a	b	c	f(c)	|b-a|
0	1	2	1.250000	-0.609375	1.000000
1	1.25	2	1.376623	-0.286264	0.750000
2	1.3766	2	1.430925	-0.117660	0.623377
3	1.4309	2	1.452402	-0.045671	0.569075
4	1.4524	2	1.460613	-0.017331	0.547598
5	1.4606	2	1.463712	-0.006520	0.539387
6	1.4637	2	1.464875	-0.002445	0.536288
7	1.4649	2	1.465310	-0.000916	0.535125
8	1.4653	2	1.465474	-0.000343	0.53469
9	1.4655	2	1.465535	-0.000128	0.534526
10	1.4655	2	1.465558	-0.000048	0.534465

Table 3.2 Solution of $x^3 - x^2 - 1 = 0$ in $[1, 2]$ using the method of false position.

5. **Computer assignment:**
   Write a computer program in FORTRAN or C that finds the zeros of a function $f(x)$ on an interval $[a, b]$ using the method of false position. Input data to the program should be a function $f(x)$, a, b, the error tolerance ϵ, and the maximum number of iterates.

   Test your program to find the root of the equation given in Exercise 1.

3.3 THE SECANT METHOD

Because the bisection and the false position methods converge at a very slow speed, our next approach is an attempt to produce a method that is faster. One such method is the **secant method**. Similar to the false position method, it is based on approximating the function by a straight line connecting two points on the graph of the function f, but we do not require f to have opposite signs at the initial points. Figure

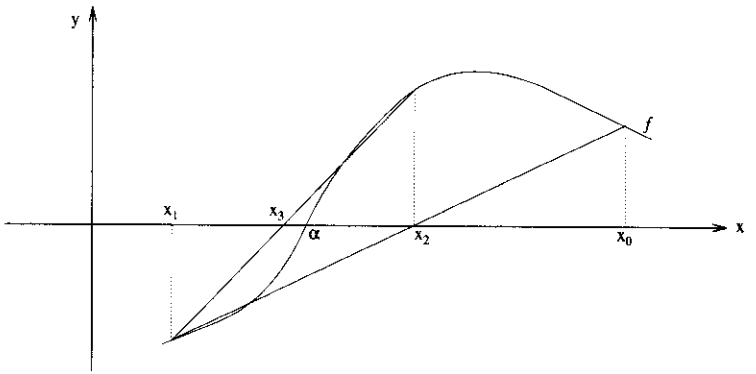


FIGURE 3.5
The secant method and the first two approximations to its zero α.

3.5 illustrates the method.

In this method, the first point, x_2, of the iteration is taken to be the point of intersection of the x-axis and the secant line connecting two starting points $(x_0, f(x_0))$ and $(x_1, f(x_1))$. The next point, x_3, is generated by the intersection of the new secant line, joining $(x_1, f(x_1))$ and $(x_2, f(x_2))$ with the x-axis. The new point, x_3, together with x_2, is used to generate the next point, x_4, and so on.

A formula for x_{n+1} is obtained by setting $x = x_{n+1}$ and $y = 0$ in the equation of the secant line from $(x_{n-1}, f(x_{n-1}))$ to $(x_n, f(x_n))$

$$x_{n+1} = x_n - f(x_n)\left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right]. \tag{3.7}$$

This is of course, the same formula as for the method of false position.

An algorithmic statement of this method is shown below.

Let x_0 and x_1 be two initial approximations,

for $n = 1, 2, \ldots, \text{ITMAX}$

$\quad x_{n+1} \leftarrow x_n - f(x_n)\left[\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}\right].$

A suitable stopping criterion is

$$|f(x_n)| \le \epsilon, \quad |x_{n+1} - x_n| \le \epsilon \quad \text{or} \quad \frac{|x_{n+1} - x_n|}{|x_{n+1}|} \le \epsilon$$

where ϵ is a specified tolerance value.

m-function 3.3

The following MATLAB function **secant.m** finds the solution of an equation using the secant method. INPUTS are a function *f*, initial approximations x_0, x_1; a tolerance *tol*, the maximum number of iterations *n*. The input function *f(x)* should be defined as an M-file.

```
function secant(f,x0,x1,tol,n)
% The secant method for solving the nonlinear equation f(x)=0.
iter=0;
u=feval(f,x0);
v=feval(f,x1);
err=abs(x1-x0);
disp('_____________________________________________________')
disp(' iter      xn       f(xn)      f(xn+1)-f(xn)      |xn+1-xn|  ')
disp('_____________________________________________________')
fprintf('%2.0f  %12.6f  %12.6f\n',iter,x0,u)
fprintf('%2.0f  %12.6f  %12.6f  %12.6f  %12.6f\n',iter,x1,v,v-u,err)
while (err>tol)&(iter<=n)&((v-u)~=0)
    x=x1-v*(x1-x0)/(v-u);
    x0=x1;
    u=v;
    x1=x;
    v=feval(f,x1);
    err=abs(x1-x0);
    iter=iter+1;
    fprintf('%2.0f  %12.6f  %12.6f  %12.6f  %12.6f\n',iter,x1,v,v-u,
err)
end
if ((v-u)==0)
    disp(' Division by zero')
end
if (iter>n)
    disp(' Method failed to converge')
end
```

EXAMPLE 3.4

Use the secant method with $x_0 = 1$ and $x_1 = 2$ to solve $x^3 - x^2 - 1 = 0$ to within 10^{-4}.

With $x_0 = 1$, $f(x_0) = -1$ and $x_1 = 2$, $f(x_1) = 3$, we have

$$x_2 = 2 - \frac{(2-1)(3)}{3-(-1)} = 1.25$$

from which $f(x_2) = f(1.25) = -0.609375$. The next iterate is

$$x_3 = 1.25 - \frac{(1.25-2)(-0.609375)}{-0.609375-3} = 1.3766234.$$

Continuing in this manner leads to the values in Table 3.3 which converge to $r = 1.4655713$.

» secant('f1',1,2,10^(-4),40)

n	xn	f(xn)	f(xn+1)-f(xn)	| xn+1-xn |
0	1.000000	-1.000000		
1	2.000000	3.000000	4.000000	1.000000
2	1.250000	-0.609375	-3.609375	0.750000
3	1.376623	-0.286264	0.323111	0.126623
4	1.488807	0.083463	0.369727	0.112184
5	1.463482	-0.007322	-0.090786	0.025325
6	1.465525	-0.000163	0.007160	0.002043
7	1.465571	3.20E-07	0.000163	0.000046

Table 3.3 Solution of $x^3 - x^2 - 1 = 0$ using the secant method with $x_0 = 1,\ x_1 = 2$.

EXERCISE SET 3.3

1. Repeat Exercise 1 of Section 3.1 using the secant method

2. Solve the equation $x^3 - 4x^2 + 2x - 8 = 0$ with an accuracy of 10^{-4} by using the secant method with $x_0 = 3, x_1 = 1$.

3. Approximate to within 10^{-6} the root of the equation $e^{-2x} - 7x = 0$ in $[1/9, 2/3]$ by the secant method.

4. Use the secant method to approximate the solution of the equation $x = x^2 - e^{-x}$ to within 10^{-5} with $x_0 = -1$ and $x_1 = 1$.

5. The function $f(x) = x^5 - 3x^3 + 2.5x - 0.6$ has two zeros in the interval $[0, 1]$. Approximate these zeros by using the secant method with:

(a) $x_0 = 0, x_1 = 1/2$.

(b) $x_0 = 3/4, x_1 = 1$.

6. If the secant method is used to find the zeros of $f(x) = x^3 - 3x^2 + 2x - 6$ with $x_0 = 1$ and $x_1 = 2$, what is x_2?

7. Use the secant method to approximate the solution of the equation $x = -e^x \sin x - 5$ to within 10^{-10} in the interval $[3, 3.5]$.

8. **Computer assignment:**
Write a computer program in FORTRAN or C that finds the zeros of a function $f(x)$ using the secant method. Input data to the program should be a function $f(x)$, two initial guesses for the root and the error tolerance ϵ.

Test your program to solve Exercise 2.

3.4 NEWTON'S METHOD

Newton's method is one of the most widely used of all iterative techniques for solving equations. Rather than using a secant line, the method uses a tangent line to the curve. Figure 3.6 gives a graphical interpretation of the method. To use the method we begin with an initial guess x_0, sufficiently close to the root α. The next approximation x_1 is given by the point at which the tangent line to f at $f(x_0, f(x_0))$ crosses the x-axis. It is clear that the value x_1 is much closer to α than the original guess x_0. If x_{n+1} denotes the value obtained by the succeeding iterations, that is the x-intercept of the tangent line to f at $(x_n, f(x_n))$, then a formula relating x_n and x_{n+1}, known as **Newton's method**, is given by

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \qquad n \geq 0 \tag{3.8}$$

provided $f'(x_n)$ is not zero.

To derive Eqn. (3.8), notice that the equation of the tangent line at $(x_0, f(x_0))$ is

$$y - f(x_0) = f'(x_0)(x - x_0).$$

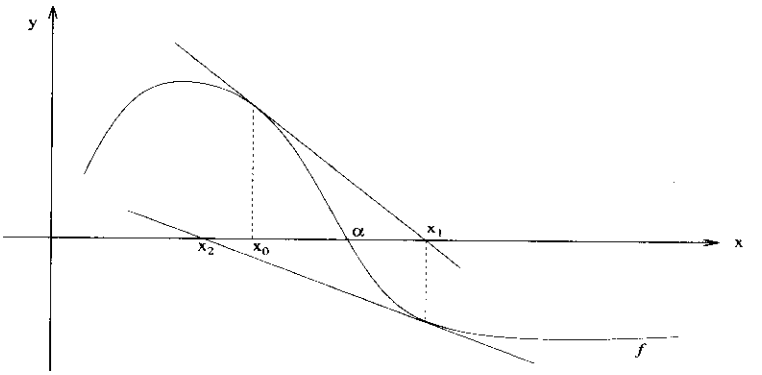


FIGURE 3.6
Newton's method and the first two approximations to its zero α.

If x_1 denotes the point where this line intersects the x-axis, then $y = 0$ at $x = x_1$, that is

$$-f(x_0) = f'(x_0)(x_1 - x_0),$$

and solving for x_1 gives

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}.$$

By repeating the process using the tangent line at $(x_1, f(x_1))$, we obtain for x_2

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

By writing the preceding equation in more general terms, one gets Eqn. (3.8). A suitable way of terminating the iterations is by using one of the stopping criteria suggested for the secant method.

An algorithmic statement of Newton's method is given below.

Given a continuously differentiable function f and an initial value x_0

for $n = 0, 1, \ldots, \text{ITMAX}$
$\quad x_{n+1} \leftarrow x_n - \frac{f(x_n)}{f'(x_n)}$

Sometimes Newton's method diverges. If x_0 is not sufficiently close to the root, the method may not converge. It is evident that one should not choose x_0 such that $f'(x_0)$ is close to zero, for then the tangent line is almost horizontal, as illustrated in Figure 3.7.

m-function 3.4

The following MATLAB function **newton.m** approximates the solution of an equation using Newton's method. INPUTS are a function *f* and its derivative *f'*; an initial guess x_0; a tolerance *tol*; the maximum number of iterations *n*. The input functions *f(x)* and *f'(x)* should be defined as M-files.

```
function newton(f,df,x0,tol,n)
% Newton's method for solving the nonlinear equation f(x)=0.
iter=0;
u=feval(f,x0);
v=feval(df,x0);
err=abs(u/v);
disp('_____________________________________________')
disp(' iter        x          f(x)        df(x)        |xn+1-xn|    ')
disp('_____________________________________________')
fprintf('%2.0f  %12.6f  %12.6f  %12.6f\n', iter, x0, u, v)
while (err>tol)&(iter<=n)&(v~=0)
    x1=x0-u/v;
    err=abs(x1-x0);
    x0=x1;
    u=feval(f,x0);
    v=feval(df,x0);
    iter=iter+1;
    fprintf('%2.0f  %12.6f  %12.6f  %12.6f  %12.6f\n',iter,x0,u,v,err)
end
if (v==0)
  disp(' division by zero')
end
if (iter>n)
  disp(' Method failed to converge')
end
```

EXAMPLE 3.5

Use Newton's method to compute a root of

$$x^3 - x^2 - 1 = 0$$

to an accuracy of 10^{-4}. Use $x_0 = 1$.

The derivative of f is $f'(x) = 3x^2 - 2x$. Using $x_0 = 1$ gives $f(1) = -1$

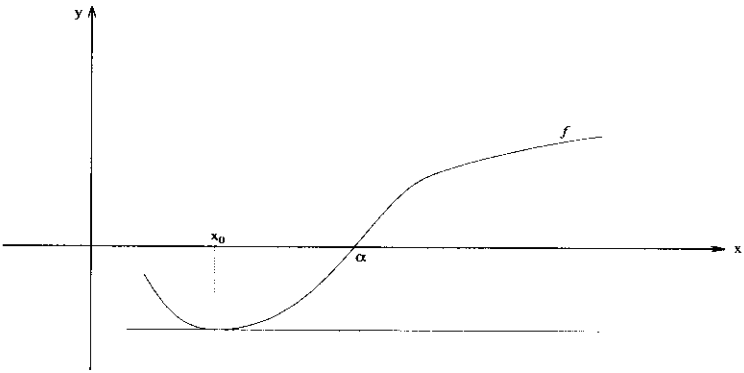


FIGURE 3.7
Failure of Newton's method due to a bad starting point.

and $f'(1) = 1$ and so the first Newton iterate is

$$x_1 = 1 - \frac{-1}{1} = 2 \qquad \text{and} \qquad f(2) = 3, \quad f'(2) = 8.$$

The next iterate is

$$x_2 = 2 - \frac{3}{8} = 1.625.$$

Continuing in this manner leads to the values in Table 3.4 which converge to $r = 1.465571$. In Table 3.4 the MATLAB function df1 is defined as follows:

```
function f=df1(x)
f=3*x.^2-2*x;
```

» newton('f1','df1',1,10^(-4),40)

iter	xn	f(xn)	f'(xn)	| xn+1-xn |
0	1.000000	-1.000000	1.000000	
1	2.000000	3.000000	8.000000	1.000000
2	1.625000	0.650391	4.671875	0.375000
3	1.485786	0.072402	3.651108	0.139214
4	1.465956	0.001352	3.515168	0.019830
5	1.465571	0.000001	3.512556	0.000385
6	1.465571	6.93E-14	3.512555	1.43E-07

Table 3.4 Solution of $x^3 - x^2 - 1 = 0$ using Newton's method with $x_0 = 1$.

EXERCISE SET 3.4

1. Find the roots of the following equations in the interval indicated with an accuracy of 10^{-3} by using Newton's method.

   (a) $x^3 + 3x^2 + x - 4 = 0$, in $[0, 1]$.

   (b) $x^4 + 2x^3 - 3x^2 - 4x - 2 = 0$, in $[1, 2]$.

   (c) $x^3 - 3x^2 + 3 = 0$, in $[1, 2]$.

2. Use Newton's method to compute a zero of the function

$$f(x) = x^5 - 3x^3 - 5x + 4$$

   to an accuracy of 10^{-6}. Use $x_0 = 1$.

3. Find an approximate root of $e^{-x} - x^2 - x = 0$ in $[-1, 1]$ using Newton's method with 10^{-5} accuracy.

4. Find the roots of the following equations correct to six decimal places using Newton's method.

   (a) $x^5 - 3x^3 - x + 4 = 0$, $x_0 = 0$.

   (b) $3x^3 - x + 4 = 0$, $x_0 = -1$.

   (c) $x^3 - x^2 + e^x = 0$, $x_0 = -1$.

   (d) $e^x + x \sin x = 0$, $x_0 = 0$.

5. Calculate an approximate value for $\sqrt{5}$ using Newton's method with $x_0 = 5$.

6. The equation $e^x + x^3 \cos x + 2 = 0$ has two positive roots in $[2, 4]$. Find them by using Newton's method.

7. Find an approximate root of $x + \sin x - \cos 2x = 0$, by using:

   (a) Newton's method with $x_0 = 0$.

   (b) The secant method with $x_0 = 0$ and $x_1 = 1$.

8. Use Newton's method with $x_0 = 0$ to approximate the value of x that produces the point on the graph of $y = x^2$ that is closest to $(3, 2)$.

9. The equation $x - \cos x = 0$ has a root near $x = 0$ and $x = 1$. Compute the root with 10^{-4} accuracy first using the secant method and then Newton's method.

10. Show that the Newton iterates for the function

$$f(x) = \frac{1}{x} - 5$$

is given by $x_{n+1} = 2x_n - 5x_n^2$. Approximate the zero of f using Newton's method with $x_0 = 1/4$.

11. Derive the formula

$$x_n = \frac{x_{n-1}^2 - b}{2x_{n-1} - a}$$

for the approximate roots of the quadratic equation

$$x^2 - ax + b = 0$$

using Newton's method. Use the formula to find the positive root of $x^2 - 13x - 1 = 0$.

12. Show that when Newton's method is used to approximate $\sqrt[3]{a}$, the sequence of iterates is given by

$$x_{n+1} = \frac{2x_n^3 + a}{3x_n^2}.$$

13. The iteration formula $x_{n+1} = x_n + ae^{-x_n} - 1$, where a is a positive constant, was obtained by applying Newton's method for some function $f(x)$. Find $f(x)$.

14. **Computer assignment:**
Write a computer program in FORTRAN or C that finds the zeros of a function $f(x)$ using Newton's method. Input data to the program should be a function $f(x)$ and its derivative f', an initial guess x_0 for the root and the error tolerance ϵ.

Test your program to solve Exercise 6.

3.5 CONVERGENCE OF THE NEWTON AND SECANT METHODS

In order to determine the order of convergence of the Newton and secant methods, we make the following definition:

DEFINITION 3.1 *Let $x_0, x_1, \ldots$ be a sequence which converges to a number α, and set $e_n = \alpha - x_n$. If there exists a number k and a positive constant C such that*

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^k} = C$$

then k is called the order of convergence of the sequence and C the asymptotic error constant.

We now examine the rate of convergence to simple roots for both the Newton and secant methods. We consider first Newton's method.

We have from Eqn. (3.8)

$$e_{n+1} = \alpha - x_{n+1} = \alpha - x_n + \frac{f(x_n)}{f'(x_n)}. \tag{3.9}$$

If f is twice-differentiable, the Taylor series expansion of f about x_n gives

$$f(\alpha) = f(x_n) + (\alpha - x_n)\, f'(x_n) + \frac{(\alpha - x_n)^2}{2} f''(\xi_n) \tag{3.10}$$

where ξ_n lies between x_n and α.

Since $f(\alpha) = 0$ and $f'(\alpha) \neq 0$, then combining Eqns. (3.9) and (3.10) we obtain

$$\begin{aligned} e_{n+1} &= -(x_n - \alpha)^2 \frac{f''(\xi_n)}{2f'(x_n)} \\ &= -e_n{}^2 \frac{f''(\xi_n)}{2f'(x_n)}. \end{aligned} \tag{3.11}$$

Therefore, if α is a simple root, that is $f'(\alpha) \neq 0$, Eqn. (3.11) shows that Newton's method converges quadratically. In other words, for a simple root, Newton's method will usually converge with the number

of accurate decimal places approximately doubling in one Newton iteration.

Similarly, It can be shown that the error for the secant method satisfies the equation

$$e_{n+1} = e_n e_{n-1} \frac{f''(\xi_n)}{2f'(\zeta_n)} \tag{3.12}$$

where ξ_n and ζ_n are in the smallest interval containing α, x_n, and x_{n-1}. Thus, the secant method is nearly as fast as Newton's method. Because the error analysis of the secant method is complicated, we omit the proof (see [9] for more details).

EXERCISE SET 3.5

1. Find the real positive root of the equation

$$x^3 - x^2 - 1 = 0$$

with an accuracy of 10^{-4} by using the secant method with $x_0 = 1$ and $x_1 = 2$. Calculate $|e_n|$ and $|e_{n+1}|/|e_n e_{n-1}|$ for $n = 2, 3, 4$ to verify the error formula (3.12).

2. Show that the sequence defined by

$$x_n = \frac{1}{2}x_{n-1} + \frac{3}{2x_{n-1}}, \quad \text{for} \quad n \geq 1$$

converges to $\sqrt{3}$ whenever the starting point $x_0 > \sqrt{3}$.

3. The function $f(x) = x^4 - x^3 - 1$ has one root in $[1, 2]$. Using Newton's method, find this zero to an accuracy of 10^{-5}. Calculate $|e_n|$ and $|e_{n+1}|/|e_n{}^2|$ for $n = 2, 3, 4$ to verify the error formula (3.11).

4. The equation $x^2 + ax + b = 0$ has two real roots, α and β. Show that the iteration method

$$x_{k+1} = -(ax_k + b)/x_k$$

is convergent near $x = \alpha$ if $|\alpha| > |\beta|$ and that

$$x_{k+1} = -b/(x_k + a)$$

is convergent near $x = \alpha$ if $|\alpha| < |\beta|$.

3.6 FIXED-POINT ITERATION

All methods discussed previously are examples of iterative methods. In this section we consider another important iterative method for the solution of equations of the form

$$g(x) = x. \tag{3.13}$$

A solution to this equation is called a **fixed point** of g. The fixed-point iteration which is used to determine the fixed point of g starts from an initial guess p_0 to determine a sequence of values $\{p_n\}$ obtained from the rule

$$p_{n+1} = g(p_n), \qquad n = 0, 1, 2, \ldots \tag{3.14}$$

If the process converges to a root α and g is continuous, then

$$\lim_{n \to \infty} p_n = \alpha$$

and a solution to Eqn. (3.13) is obtained.

Geometrically, a fixed point p of g can be regarded as the point of intersection of the functions $y = x$ and $y = g(x)$. Figures 3.8 and 3.9 give graphical interpretations of convergence and divergence iterations, respectively.

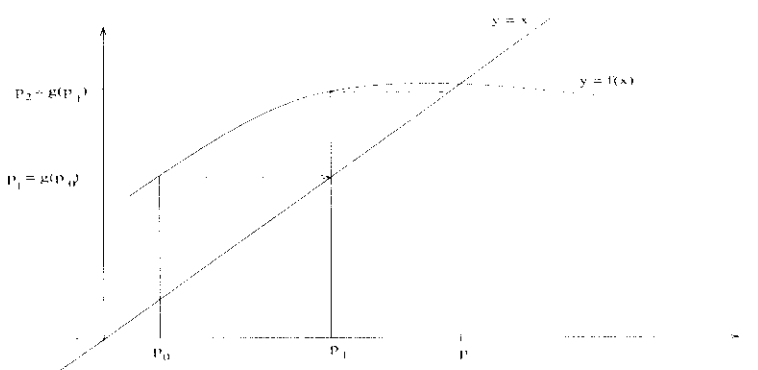


FIGURE 3.8
The fixed-point iteration with convergence iteration to its zero p.

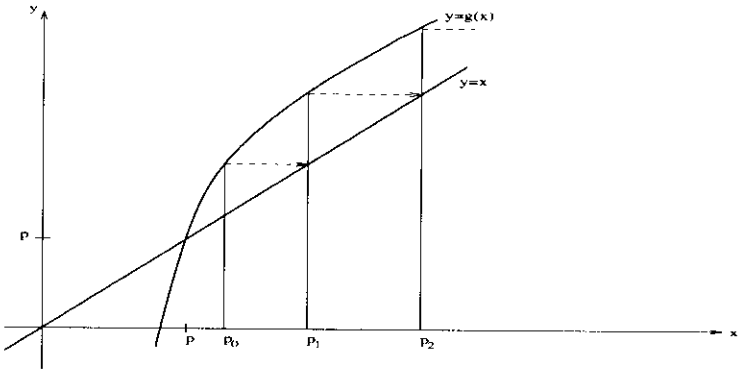


FIGURE 3.9
The fixed-point iteration with divergence iteration from its zero p .

EXAMPLE 3.6

Solve the equation $0.4-0.1x^2 = 0$ with $x_0 = 1$ using the fixed-point iteration and choosing

(F1) $g(x) = 0.4 + x - 0.1x^2$,

(F2) $g(x) = \frac{4}{x}$.

Table 3.5 gives the iterates x_n of the fixed-point iteration method for both F1 and F2. The exact solution is 2.0. From the results given in Table 3.5, it is interesting to note that the choice of $g(x)$ in F1 led to convergence and the one in F2 to divergence.

The question that one can ask is how can we choose $g(x)$ so that convergence will occur? The following Theorem gives an answer to that question.

THEOREM 3.2

If g is continuous on $[a, b]$ and $a \leq g(x) \leq b$ for all x in $[a, b]$, then g has at least a fixed point in $[a, b]$.

Further, suppose $g'(x)$ is continuous on (a, b) and that a positive constant c exists with

$$|g'(x)| \leq c < 1, \qquad \textit{for all } x \textit{ in } (a, b). \tag{3.15}$$

Then there is a unique fixed point α of g in $[a, b]$. Also, the iterates

$$x_{n+1} = g(x_n) \qquad n \geq 0$$

will converge to α for any choice of x_0 in $[a, b]$.

n	F1	F2
0	1.00000000	1
1	1.30000000	4
2	1.53100000	1
3	1.69660390	4
4	1.80875742	1
5	1.88159708	4
6	1.92755632	1
7	1.95600899	4
8	1.97341187	1
9	1.98397643	4
10	1.99036018	1
11	1.99420682	4
12	1.99652073	1
13	1.99791123	4
14	1.99874630	1
15	1.99924762	4
16	1.99954852	1
17	1.99972909	4
18	1.99983745	1

Table 3.5 The fixed-point iteration for (F1) and (F2).

Proof: Define the function $f(x) = x - g(x)$. Then f is continuous on $[a, b]$, and $f(a) \leq 0$ and $f(b) \geq 0$. By the Intermediate Value Theorem there exists α in (a, b) for which $f(x) = 0$. Thus, α is a fixed point of g.

In addition, by the Mean Value Theorem, we have that for any two fixed points r and s in $[a, b]$ there exists ξ such that

$$|r - s| = |g(r) - g(s)| = |g'(\xi)||r - s| \leq c|r - s|.$$

Thus,

$$|r - s| \leq c|r - s| \quad \text{or} \quad (1 - c)|r - s| \leq 0.$$

Since $c < 1$, we must have $r = s$. Hence, g has a unique fixed point in $[a, b]$.

For the convergence,

$$|\alpha - x_{n+1}| = |g(\alpha) - g(x_n)| = |g'(\xi_n)||\alpha - x_n| \leq c|\alpha - x_n|$$

with $\xi_n \in (\alpha, x_n)$. By induction

$$|\alpha - x_n| \leq c^n |\alpha - x_0| \qquad n \geq 0.$$

As $n \to \infty$, $c^n \to 0$; therefore, $x_n \to \alpha$.

EXAMPLE 3.7

Suppose we want to solve the equation $x = \pi + 0.5 \sin x$ using the fixed-point iteration. If $g(x) = \pi + 0.5 \sin x$, let us find the interval $[a, b]$ satisfying the hypotheses of Theorem 3.2.

It is easy to see that g is continuous and

$$0 \leq g(x) \leq 2\pi \qquad \text{for all } x \text{ in } [0, 2\pi].$$

Moreover, g' is continuous and

$$|g'(x)| = |0.5 \cos x| < 1 \quad \text{in } [0, 2\pi],$$

thus, g satisfies the hypotheses of Theorem 3.2 and has a unique fixed point in the interval $[0, 2\pi]$.

EXAMPLE 3.8

The equation $x = \frac{5}{x^2} + 2$ has exactly one zero in $[2.5, 3]$. Use the fixed-point iteration method to approximate the root with $g(x) = 5/x^2 + 2$ and $x_0 = 2.5$ to within 10^{-4}.

Both $g(x)$ and $g'(x) = -10/x^3$ are continuous in $[2.5, 3]$. It is easy to check that $|g'(x)| \leq 1$ and $2.5 \leq g(x) \leq 3$ for all x in $[2.5, 3]$. Thus g has a unique fixed-point in $[2.5, 3]$.

The first three iterations are

$$\begin{aligned} p_1 &= 5/p_0^2 - 2 = 2.8 \\ p_2 &= 5/p_1^2 - 2 = 2.637755 \\ p_3 &= 5/P_2^2 - 2 = 2.718623. \end{aligned}$$

Continuing in this manner leads to the values in Table 3.6 which converge to $r = 2.69062937$.

m-function 3.5

The following MATLAB function **fixed.m** finds the solution of an equation using the fixed-point iteration method. INPUTS are a function *g*; an initial approximation x_0; a tolerance *tol*; the maximum number of iterations *n*. The input function *g(x)* should be defined as an M-file.

```
function fixed(g,x0,tol,n)
% The fixed-point iteration method for solving the nonlinear
%  equation f(x)=0.
iter=0;
u=feval(g,x0);
err=abs(u-x0);
disp('_____________________________________________________')
disp('   iter              x              g(x)              |xn+1-xn|  ')
disp('_____________________________________________________')
fprintf('%2.0f  %12.6f  %12.6f\n',iter,x0,u)
while (err>tol)&(iter<=n)
   x1=u;
   err=abs(x1-x0);
   x0=x1;
   u=feval(g,x0);
   iter=iter+1;
   fprintf('%2.0f  %12.6f  %12.6f  %12.8f\n',iter,x0,u,err)
end
if (iter>n)
   disp('  Method failed to converge')
end
```

In Table 3.6 the MATLAB function g1 is defined as follows:

```
function f=g1(x);
f=5/x.^2+2;
```

EXERCISE SET 3.6

1. Solve the following equations using the fixed-point iteration method:

   (a) $x = \sin x + x + 1$, in $[3.5, 5]$.

» fixed('g1',2.5,10^(-4),100)

iter	xn	g(xn)	| xn+1-xn |
0	2.500000	2.800000	
1	2.800000	2.637755	0.30000000
2	2.637755	2.718623	0.16224490
3	2.718623	2.676507	0.08086781
4	2.676507	2.697965	0.04211628
5	2.697965	2.686906	0.02145790
6	2.686906	2.692572	0.01105819
7	2.692572	2.689660	0.00566568
8	2.689660	2.691154	0.00291154
9	2.691154	2.690387	0.00149391
10	2.690387	2.690781	0.00076713
11	2.690781	2.690579	0.00039377
12	2.690579	2.690683	0.00020216
13	2.690683	2.690629	0.00010378
14	2.690629	2.690657	0.00005328

Table 3.6 Solution of $x = \frac{5}{x^2} + 2$ using the fixed-point iteration method with $x_0 = 2.5$.

(b) $x = \sqrt{x^2+1} - x + 1$, in $[0, 3]$.

(c) $x = \ln x^2 + x - 2$, in $[-4, -2]$.

2. For each of the following functions, locate an interval on which fixed-point iteration will converge.

   (a) $x = 0.2 \sin x + 1$.

   (b) $x = 1 - x^2/4$.

3. Find the solution of the following equations using the fixed-point iteration:

   (a) $x = x^5 - 0.25$ starting from $x_0 = 0$.

   (b) $x = 2 \sin x$ starting from $x_0 = 2$.

   (c) $x = \sqrt{3x+1}$ starting from $x_0 = 2$.

   (d) $x = \frac{2-e^x+x^2}{3}$ starting from $x_0 = 1$.

4. Use the fixed-point iteration to find an approximation to $\sqrt{2}$ with 10^{-3} accuracy.

5. The quadratic equation $x^2 - 2x - 3 = 0$ has two roots. Consider the following rearrangements to approximate the roots using the fixed-point iteration:

   (a) $x = \sqrt{2x+3}$.

   (b) $x = 3/(x-2)$.

   (c) $x = (x^2 - 3)/2$.

   starting from $x_0 = 4$.

6. Let $g(x) = \frac{x+4}{2}$

   (a) Show that $\alpha = 4$ is a fixed point of $g(x)$.

   (b) Let $x_0 = 5$, show that $|\alpha - x_n| = |\alpha - x_0|/2^n$ for $n = 1, 2, \ldots$.

3.7 MULTIPLE ROOTS AND THE MODIFIED NEWTON METHOD

We say that α is a root of **multiplicity** k of f if

$$f(\alpha) = f'(\alpha) = \cdots = f^{(k-1)} = 0, \qquad \text{but} \qquad f^k(\alpha) \neq 0. \tag{3.16}$$

In Section 3.4, Newton's algorithm was derived under the assumption that $f'(\alpha) \neq 0$, that is, α was assumed to be a simple zero. As one can see from Example 3.9 below, difficulties might occur with Newton's method if α is not a simple root.

EXAMPLE 3.9

The function $f(x) = x^3 - 7x^2 + 11x - 5$ has a root of multiplicity two at $x = 1$. Use Newton's method with $x_0 = 0$ to approximate it.

A plot of $f(x)$ from MATLAB is shown in Figure 3.10.

From Definition 3.1 and the numerical results given in Table 3.7, it shows that Newton's method convergence linearly to $a = 1$ since the

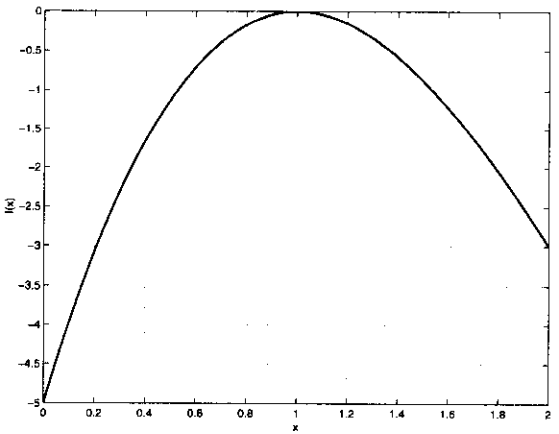


FIGURE 3.10
Plot of $f(x) = x^3 - 7x^2 + 11x - 5$ with a root of multiplicity two at $x = 1$.

error in each successive iteration is proportional to the error in the previous iteration. That is,

$$|a - x_{n+1}|/|a - x_n| \approx C,$$

where $C \approx 1/2$ as shown in Table 3.7. The difficulty with multiple roots leads us to consider a modification of Newton's method.

As before, we wish to find a zero of a function f. We assume that f has a zero α of multiplicity m. Let us define the new function

$$u(x) = \frac{f(x)}{f'(x)}. \tag{3.17}$$

Using Eqn. (3.16), the Taylor series expansion of $f(x)$ about α becomes

$$f(x) = (x - \alpha)^m h(x) \tag{3.18}$$

with

$$h(x) = \frac{f^{(m)}(\alpha)}{m!} + \frac{f^{(m+1)}(\alpha)}{(m+1)!}(x - \alpha) + \cdots$$

By differentiating Eqn. (3.18) we obtain

$$\begin{aligned} f'(x) &= (x - \alpha)^m h'(x) + m(x - \alpha)^{m-1} h(x) \\ &= (x - \alpha)^{m-1} \left[(x - \alpha) h'(x) + m h(x) \right]. \end{aligned} \tag{3.19}$$

n	x_n	$\lvert x_{n+1}-x_n\rvert$	$\lvert a-x_n\rvert$	$\lvert e_{n+1}\rvert / \lvert e_n\rvert$
0	0.000000	0.454545	1.000000	0.545455
1	0.454545	0.257290	0.545455	0.528302
2	0.711835	0.139399	0.288165	0.516254
3	0.851234	0.073073	0.148766	0.508807
4	0.924307	0.037498	0.075693	0.504600
5	0.961805	0.019008	0.038195	0.502353
6	0.980813	0.009571	0.019187	0.501191
7	0.990384	0.004802	0.009616	0.500599
8	0.995186	0.002406	0.004814	0.500300
9	0.997592	0.001204	0.002408	0.500150
10	0.998795	0.000602	0.001205	0.500075
11	0.999398	0.000301	0.000602	0.500038
12	0.999699	0.000151	0.000301	0.500019
13	0.999849	0.000075	0.000151	0.500009
14	0.999925	0.000000	0.000075	

Table 3.7 Solution of $x^3 - 7x^2 + 11x - 5 = 0$ using Newton's method with $x_0 = 0$.

Thus from (3.18) and (3.19), (3.17) can be written in the form

$$u(x) = \frac{(x-\alpha)h(x)}{(x-\alpha)h'(x) + mh(x)} = (x-\alpha)\Psi(x) \tag{3.20}$$

where

$$\Psi(x) = \frac{h(x)}{(x-\alpha)h'(x) + mh(x)}.$$

Since $h(\alpha) = \frac{f^{(m)}(\alpha)}{m!}$, we have

$$\lim_{x\to\alpha} \Psi(x) = \frac{1}{m} \tag{3.21}$$

and

$$\lim_{x\to\alpha} u'(x) = \lim_{x\to\alpha} \left[(x-\alpha)\Psi'(x) + \Psi(x)\right] = \frac{1}{m}. \tag{3.22}$$

In view of (3.20) and (3.22), we conclude that $u(x)$ has a simple root at $x = \alpha$. Applying now Newton's method to the function $u(x)$ and using (3.22) yields

$$x_{n+1} = x_n - m\frac{f(x_n)}{f'(x_n)} \tag{3.23}$$

where m represents the multiplicity of the root α. Eqn. (3.23) is called the **modified Newton's method**.

m-function 3.6

The following MATLAB function **newton2.m** finds the solution of an equation using the modified Newton's method. INPUTS are a function *f* and its derivative *f'*; an initial approximation x_0; the number of multiplicity *m*; a tolerance *tol*; the maximum number of iterations *n*. The input functions *f(x)* and *f'(x)* should be defined as M-files.

```
function newton2(f,df,x0,m,tol,n)
% Modified Newton's method for solving the nonlinear equation
% f(x)=0.
iter=0;
u=feval(f,x0);
v=feval(df,x0);
err=abs(m*u/v);
disp('_____________________________________________')
disp(' iter          x          f(x)          df(x)          |xn+1-xn|  ')
disp('_____________________________________________')
fprintf('%2.0f  %12.6f  %12.6f  %12.6f\n',iter,x0,u,v)
while (err>tol)&(iter<=n)&(v~=0)
  x1=x0-m*u/v;
  err=abs(x1-x0);
  x0=x1;
  u=feval(f,x0);
  v=feval(df,x0);
  iter=iter+1;
  fprintf('%2.0f  %12.6f  %12.6f  %12.6f  %12.7f\n',iter,x0,u,v,err)
  pause(1)
  y(iter)=err;
end
if (v==0)
  disp(' division by zero')
end
if (iter>n)
  disp(' Method failed to converge')
end
```

EXAMPLE 3.10

Use the modified Newton's method with $x_0 = 0$ and $m = 2$ to approximate the double root $a = 1$ of the equation given in Example 3.9. Use the results to show that the modified Newton's method converges quadratically.

n	x_n	$\lvert x_{n+1}-x_n\rvert$	$\lvert a-x_n\rvert$	$\lvert e_{n+1}\rvert / \lvert e_n\rvert^2$
0	0.00000000	0.90909091	1.00000000	0.09090909
1	0.90909091	0.08991009	0.09090909	0.12087912
2	0.99900100	0.00099888	0.00099900	0.12495319
3	0.99999988	0.00000012	0.00000012	
4	1.00000000	0.00000000	0.00000000	

Table 3.8 Solution of $x^3 - 7x^2 + 11x - 5 = 0$ using the modified Newton's method with $x_0 = 0$ and $m = 2$.

Using the modified Newton's method

$$x_{n+1} = x_n - 2\frac{x_n^3 - 7x_n^2 + 11x_n - 5}{3x_n^2 - 14x_n + 11}$$

with $m = 2$, we get the results shown in Table 3.8. From the numerical results given in Table 3.8, it shows that the modified Newton's method converges quadratically to $a = 1$ since the error in each successive iteration is proportional to the square of the error in the previous iteration. That is,

$$|a - x_{n+1}|/|a - x_n|^2 \approx C,$$

where $C \approx 0.12$.

Note that it took 14 iterations for Newton's method to converge and only 3 iterations for the modified Newton's method with the same tolerance.

EXAMPLE 3.11

The equation $x^3 - 4x^2 - 3x + 18 = 0$ has a double root at $x = 3$. Use the modified Newton's method with $x_0 = 0$ and $m = 2$ to approximate it.

Define the MATLAB function f3 and its derivative df3 as follows:

```
function f=f3(x)
f=x.^3-4*x.^2-3*x+18;

function f=df3(x);
f=3*x.^2-8*x-3;
```

» newton2('f3','df3',0,2,10^(-12),40)

iter	xn	f(xn)	f '(xn)	| xn+1-xn |
0	0.000000	18.000000	-3.000000	---
1	12.000000	1134.000000	333.000000	12.000000
2	5.189189	34.454544	36.269540	6.810811
3	3.289273	0.442600	3.143766	1.899916
4	3.007700	0.000297	0.077175	0.281573
5	3.000006	1.70E-10	5.90E-05	0.007694
6	3.000000	0	0	5.90E-06
7	3.000000	0	0	0

Table 3.9 Solution of $x^3 - 4x^2 - 3x + 18 = 0$ using the modified Newton's method with $x_0 = 0$ and $m = 2$.

and then call the function newton2 to get the results shown in Table 3.9.

EXERCISE SET 3.7

1. The function
$$f(x) = x^3 + 2x^2 - 7x + 4$$
has a zero of order 2 at $x = 1$. Start with $x_0 = 2$ and use Newton's method to find this zero. Show that in this case Newton's method converges linearly. Also show that if the modified Newton's method is used with $x_0 = 2$ and $m = 2$ to find this zero, the iteration does converge quadratically.

2. Show that the function
$$f(x) = (4x - 8) + x^3 - 2x^2 - 4(x^2 - 2x)$$
has a zero of order 3. Use the modified Newton method to find this zero starting with $x_0 = 1$.

3. Try to solve the equation $x^2 - 1 = 0$ using Newton's method with $x_0 = 0$. Use a graphic interpretation to explain what happened.

4. **Computer assignment:**
   Repeat the computer assignment of Section 3.4 for the modified Newton's method.

   Test your program to solve Exercise 1.

3.8 NEWTON'S METHOD FOR NONLINEAR SYSTEMS

Newton's method introduced in Section 3.4 for finding the zeros of a nonlinear function of a single variable, can be extended to nonlinear functions in several variables. However, the problem of finding the zeros of nonlinear functions is much more difficult than for linear equations. Starting with an initial estimate of the solution, Newton's method consists of approximating each nonlinear function by its tangent plane; and the common root of the resulting linear equations provides the next approximation. To see how the algorithm works, we first consider a system of two nonlinear equations. That is,

$$f_1(x_1, x_2) = 0$$
$$f_2(x_1, x_2) = 0$$

Let $(x_1^{(0)}, x_2^{(0)})$ be an initial estimate to the solution and f be differentiable at $\mathbf{x}^{(0)} = (x_1^{(0)}, x_2^{(0)})$. The equation of the tangent plane to the function $y = f_1(x_1, x_2)$ at $(x_1^{(0)}, x_2^{(0)})$ is

$$y - f_1(x_1^{(0)}, x_2^{(0)}) = \frac{\partial}{\partial x_1}\left[f_1(x_1^{(0)}, x_2^{(0)})\right](x_1 - x_1^{(0)}) + \frac{\partial}{\partial x_2}\left[f_1(x_1^{(0)}, x_2^{(0)})\right](x_2 - x_2^{(0)})$$

Similarly, the equation of the tangent plane to the function $y = f_2(x_1, x_2)$ at $(x_1^{(0)}, x_2^{(0)})$ is

$$y - f_2(x_1^{(0)}, x_2^{(0)}) = \frac{\partial}{\partial x_1}\left[f_2(x_1^{(0)}, x_2^{(0)})\right](x_1 - x_1^{(0)}) + \frac{\partial}{\partial x_2}\left[f_2(x_1^{(0)}, x_2^{(0)})\right](x_2 - x_2^{(0)})$$

The intersection of these two tangent planes with the xy-plane is obtained by solving the system

$$\begin{aligned}\frac{\partial}{\partial x_1}\left[f_1(x_1^{(0)},x_2^{(0)})\right]\Delta x_1^{(0)}+\frac{\partial}{\partial x_2}\left[f_1(x_1^{(0)},x_2^{(0)})\right]\Delta x_2^{(0)}&=-f_1(x_1^{(0)},x_2^{(0)})\\ \frac{\partial}{\partial x_1}\left[f_2(x_1^{(0)},x_2^{(0)})\right]\Delta x_1^{(0)}+\frac{\partial}{\partial x_2}\left[f_2(x_1^{(0)},x_2^{(0)})\right]\Delta x_2^{(0)}&=-f_2(x_1^{(0)},x_2^{(0)})\end{aligned} \tag{3.24}$$

where $\Delta x_1^{(0)} = x_1 - x_1^{(0)}$ and $\Delta x_2^{(0)} = x_2 - x_2^{(0)}$.

We solve Eqn. (3.24) by Gaussian elimination (see Chapter 4) and the resulting solution $(x_1^{(0)}, x_2^{(0)})$ is used to get an improved estimate to the solution. That is,

$$\begin{aligned}x_1^{(1)} &= x_1^{(0)} + \Delta x_1^{(0)}\\ x_2^{(1)} &= x_2^{(0)} + \Delta x_2^{(0)}\end{aligned}$$

We repeat the process until f_1 and f_2 are close to zero or $\|\Delta\mathbf{x}\|_\infty = \max_i\{\,|\Delta x_i|\,\}$ is less than a given specified tolerance value, in which case convergence has been achieved.

Starting with an approximate solution $(x_1^{(i)}, x_2^{(i)})$, the vector-matrix notation of these equations is

$$\begin{bmatrix}\partial f_1(x_1^{(i)},x_2^{(i)})/\partial x_1 & \partial f_1(x_1^{(i)},x_2^{(i)})/\partial x_2\\ \partial f_2(x_1^{(i)},x_2^{(i)})/\partial x_1 & \partial f_2(x_1^{(i)},x_2^{(i)})/\partial x_2\end{bmatrix}\begin{bmatrix}\Delta x_1^{(i)}\\ \Delta x_2^{(i)}\end{bmatrix} = -\begin{bmatrix}f_1(x_1^{(i)},x_2^{(i)})\\ f_2(x_1^{(i)},x_2^{(i)})\end{bmatrix} \tag{3.25}$$

and

$$\begin{bmatrix}x_1^{(i+1)}\\ x_2^{(i+1)}\end{bmatrix} = \begin{bmatrix}x_1^{(i)}+\Delta x_1^{(i)}\\ x_2^{(i)}+\Delta x_2^{(i)}\end{bmatrix}, \quad i = 0,1,2,\ldots$$

For a system of n equations, we can write Newton's method by generalizing Eqn. (3.25). The generalized system is given by

$$\begin{bmatrix}\frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n}\\ \vdots & \cdots & \vdots\\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n}\end{bmatrix}\begin{bmatrix}\Delta x_1^{(i)}\\ \vdots\\ \Delta x_n^{(i)}\end{bmatrix} = -\begin{bmatrix}f_1\\ \vdots\\ f_n\end{bmatrix} \tag{3.26}$$

where the functions are evaluated at $\mathbf{x}^{(i)} = (x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)})$. Solve this system to compute

$$\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \Delta\mathbf{x}^{(i)}, \quad i = 0,1,2,\ldots. \tag{3.27}$$

The coefficient matrix in Eqn. (3.26) is known as the Jacobian matrix $\mathbf{J}(x_1, \ldots, x_n)$. If the initial estimate is sufficiently close to the solution, Newton's method is generally expected to give quadratic convergence, provided the Jacobian matrix is nonsingular.

We now illustrate the method on the following 2×2 system.

EXAMPLE 3.12

Solve the nonlinear system

$$f_1(x_1, x_2) = x_1^3 + 3x_2^2 - 21 = 0$$
$$f_2(x_1, x_2) = x_1^2 + 2x_2 + 2 = 0$$

by Newton's method starting, with the initial estimate $\mathbf{x}^{(0)} = (1, -1)$. Iterate until $\|\Delta\mathbf{x}\|_\infty = \max_i\{\,|\Delta x_i|\,\} < 10^{-6}$.

The Jacobian matrix is

$$\mathbf{J}(x_1, x_2) = \begin{bmatrix} 3x_1^2 & 6x_2 \\ 2x_1 & 2 \end{bmatrix}$$

At the point $(1, -1)$ the function vector and the Jacobian matrix take on the values

$$\mathbf{F}(1, -1) = \begin{bmatrix} -17 \\ 1 \end{bmatrix}, \quad \mathbf{J}(1, -1) = \begin{bmatrix} 3 & -6 \\ 2 & 2 \end{bmatrix}$$

the differentials $\Delta x_1^{(0)}$ and $\Delta x_2^{(0)}$ are solutions of the system

$$\begin{bmatrix} 3 & -6 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} = -\begin{bmatrix} -17 \\ 1 \end{bmatrix}.$$

Its solution is

$$\Delta\mathbf{x}^{(0)} = \begin{bmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \end{bmatrix} = \begin{bmatrix} 1.555556 \\ -2.055560 \end{bmatrix}.$$

Thus, the next point of iteration is

$$\mathbf{x}^{(1)} = \begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1.555556 \\ -2.055556 \end{bmatrix} = \begin{bmatrix} 2.555556 \\ -3.055556 \end{bmatrix}.$$

Similarly, the next three points are

$$\mathbf{x}^{(2)} = \begin{bmatrix} 1.865049 \\ -2.500801 \end{bmatrix}, \quad \mathbf{x}^{(3)} = \begin{bmatrix} 1.661337 \\ -2.359271 \end{bmatrix}, \quad \text{and} \quad \mathbf{x}^{(4)} = \begin{bmatrix} 1.643173 \\ -2.349844 \end{bmatrix}.$$

The results are summarized in Table 3.10.

Iteration i	$\mathbf{x}^{(i)}$	$\|\Delta\mathbf{x}\|_\infty$
0	$[1, -1]$	
1	$[2.555556, -3.055556]$	2.055556
2	$[1.865049, -2.500801]$	0.690 507
3	$[1.661337, -2.359271]$	0.203712
4	$[1.643173, -2.349844]$	0.018164
5	$[1.643038, -2.349787]$	0.000135
6	$[1.643038, -2.349787]$	7.3×10^{-9}

Table 3.10 Newton's method for Example 3.12.

This example shows that Newton's method can converge very rapidly once an initial estimate sufficiently close to the solution is obtained. A disadvantage of Newton's method is the necessity of solving the linear system (3.26) at each step.

To solve the above example using the MATLAB function newton_sys first requires saving the nonlinear system and its related Jacobian matrix, respectively, as M-files:

```
function f=f1(X)
x=X(1);y=X(2);
f=[x.^3+3*y.^2-21; x.^2+2*y+2];

function df=df1(X)
x=X(1);y=X(2);
df=[3*x.^2, 6*y; 2*x, 2];
```

Now call the MATLAB function

```
>> newton_sys('f1', 'df1', [1,-1]', 10^(8), 40)
```

to get the results shown in Table 3.10.

m-function 3.8

The following MATLAB function **newton_sys.m** finds the solution of a nonlinear system of equations using Newton's method. INPUTS are the system *F* given by an *(nx1)* vector; the Jacobian *JF* of *F* given by an *(nxn)* matrix; the initial vector *xx0*; a tolerance *tol*; the maxim number of iterations *maxit*. The system *F* and its Jacobian *JF* should be defined as M-files.

```
function newton_sys(F,JF,xx0,tol,maxit)
% Solve a nonlinear system of equations using Newton's method.
x0=xx0
iter=1;
while(iter<=maxit)
  y=-feval(JF,x0)\feval(F,x0);
  xn=x0+y;
  err= max(abs(xn-x0));
  if (err<=tol)
    x=xn;
    fprintf('  Newton's method converges after %3.0f iterations to
\n', iter)
    x
    return;
  else
    x0=xn;
  end
  iter=iter+1;
end
disp('  Newton's method does not converge')
x=xn;
```

EXERCISE SET 3.8

1. Find the Jacobian matrix $J(x, y)$ at the point $(-1, 4)$ for the functions

$$f_1(x, y) = x^3 - y^2 + y = 0,$$
$$f_2(x, y) = xy + x^2 = 0.$$

2. Find the exact solution for the nonlinear system

$$x^2 + y^2 = 4,$$
$$x^2 - y^2 = 1.$$

Solve the following nonlinear systems using Newton's method

3. $f_1(x, y) = 3x - y - 3 = 0,$
   $f_2(x, y) = x - y + 2 = 0.$

4. $f_1(x, y) = x^2 + 4y^2 - 16 = 0,$
   $f_2(x, y) = x^2 - 2x - y + 1 = 0.$

5. $f_1(x, y, z) = x^2 + y^2 + z^2 - 9 = 0,$
   $f_2(x, y, z) = xyz - 1 = 0,$
   $f_3(x, y, z) = x + y - z^2 = 0.$

APPLIED PROBLEMS FOR CHAPTER 3

1. A relationship for the compressibility factor c of real gases has the form

$$c = \frac{1 + x + x^2 - x^3}{(1 - x)^3}$$

   If $c = 0.9$, find the value of x in the interval $[-0.1, 0]$ using the bisection method.

2. The temperature in the interior of a material with imbedded heat sources is obtained from the solution of the equation

$$e^{-(1/2)t} \cosh^{-1}(e^{(1/2)t}) = \sqrt{k/2}$$

   Given that $k = 0.67$, find the temperature t.

3. L.L. Vant-Hull (*Solar Energy*, **18**,33 (1976)) derived the following equation for the geometrical concentration factor, C, in the study of solar energy

$$C = \frac{\pi(h/\cos A)^2 F}{0.5\pi D^2(1 + \sin A - 0.5\cos A)}$$

   Given $h = 290$, $C = 1100$, $F = 0.7$, and $D = 13$, what is the value of A?

4. The Legendre polynomial of sixth order is given by

$$P_6(x) = \frac{1}{48}(693x^6 - 945x^4 + 315x^2 - 15)$$

Find the zeros of $P_6(x)$ using Newton's method. (*Note*: All the zeros of the Legendre polynomial are less than one in magnitude and, for polynomials of even order, are symmetric about the origin.)

5. The Chebyshev polynomial of degree 6 is given by

$$T_6(x) = 32x^6 - 48^4 + 18x^2 - 1$$

Find the zeros of $T_6(x)$ using Newton's method.
(*Note*: All the zeros of the Chebyshev polynomial are less than one in magnitude.)

6. In the study of a natural gas reservoir, Collier et al. (*Soc. Petr. Engrs.*, **21**, 5 (1981)) derived the following equation that governs the relationship of the gas pressure and reservoir volume:

$$x = y^{-1.5}\left[\frac{y - 1/(1+k)}{1 - 1/(1+k)}\right]^b \qquad \text{where}$$

$b = 1.5k/(1+k)$
$x =$ normalized volume, dimensionless
$y =$ normalized pressure, dimensionless
$k = 1.06315$, dimensionless

A reservoir will be useful until the pressure drops to a value that makes production uneconomic. Determine the value of y where the solution curve intersects the cutoff line $x = 0.15$.
Note: Try the secant method with $y_0 = 1$ and $y_1 = 2$.

Chapter 4

System of Linear Equations

Linear systems of equations arise in many problems in engineering and science, as well as in mathematics, such as the study of numerical solution of boundary-value problems and partial differential equations.

Solution algorithms for these types of problems may be either direct or iterative. In direct methods, the solution is obtained in a fixed number of steps subject only to rounding errors, while iterative methods are based on successive improvement of initial guesses for the solution.

In this chapter, both techniques will be considered to solve the linear system

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & b_2 \\ \vdots & & \vdots & & \vdots & & \vdots & = & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & b_n \end{array} \tag{4.1}$$

for the unknowns $x_1, x_2, \ldots, x_n$, given the coefficients $a_{ij},\ i, j = 1, 2, \ldots, n$ and the constants $b_i,\ i = 1, 2, \ldots, n$.

4.1 MATRICES AND MATRIX OPERATIONS

Before studying linear systems of equations, it is useful to consider some algebra associated with matrices.

DEFINITION 4.1 *An n by m matrix is a rectangular array of real or complex numbers that can be written as*

$$\mathbf{A} = (a_{ij}) = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

a_{ij} denotes the *element* or *entry* that occurs in row i and column j of A, and the *size* of a matrix is described by specifying the number of rows and columns that occur in the matrix. If a matrix has only one row, we call it a *row vector*, and a matrix having only one column is called a *column vector*. Capital letters will be used to denote matrices and lowercase letters to denote their entries.

A matrix A with n rows and n columns is called a *square matrix* of order n, and the elements $a_{11}, a_{22}, \dots, a_{nn}$ are called the *main diagonal* of A.

EXAMPLE 4.1

Let

$$A = \begin{bmatrix} -1 & 3 & 4 & 0 \\ 5 & -6 & 2 & 7 \\ 3 & 0 & 1 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -3 & 4 & 2 \end{bmatrix}, \quad C = \begin{bmatrix} \pi \\ 3 \\ 0 \end{bmatrix}$$

In these examples, A is a 3 by 4 (written 3×4) matrix, B has only one row and is a row vector, and C has only one column and is a column vector.

The following definition gives the basic operations on matrices.

DEFINITION 4.2

i) If A and B are two matrices of order $n \times m$, then the sum of A and B is the $n \times m$ matrix $C = A + B$ whose entries are

$$c_{ij} = a_{ij} + b_{ij}$$

ii) If A is a matrix of order $n \times m$ and λ a real number, then the product of λ and A is the $n \times m$ matrix $C = \lambda A$ whose entries are

$$c_{ij} = \lambda a_{ij}$$

iii) If A is a matrix of order $n \times m$ and B is a matrix of order $m \times p$, then the matrix product of A and B is the $n \times p$ matrix $C = AB$ whose entries are

$$c_{ij} = \sum_{k=1}^{m} a_{ik} b_{kj}$$

iv) If A is a matrix of order $n \times m$, then the transpose of A is the $m \times n$ matrix $C = A^T$ whose entries are

$$c_{ij} = a_{ji}$$

EXAMPLE 4.2

If

$$A = \begin{bmatrix} 1 & -2 & 4 \\ 3 & 5 & -6 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 3 & -7 \\ 1 & -3 & 8 \end{bmatrix}$$

---

then

$$A + B = \begin{bmatrix} 1 & 1 & -3 \\ 4 & 2 & 2 \end{bmatrix}, \quad -3B = \begin{bmatrix} 0 & -9 & 21 \\ -3 & 9 & -24 \end{bmatrix},$$

$$A - B = \begin{bmatrix} 1 & -5 & 11 \\ 2 & 8 & -14 \end{bmatrix}, \quad A^T = \begin{bmatrix} 1 & 3 \\ -2 & 5 \\ 4 & -6 \end{bmatrix}$$

if

$$A = \begin{bmatrix} 1 & 3 & -1 \\ 2 & 0 & -1 \\ 4 & 2 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 1 \\ -3 & 2 \\ 3 & 0 \end{bmatrix}$$

then

$$AB = \begin{bmatrix} -10 & 7 \\ 1 & 2 \\ -4 & 8 \end{bmatrix}, \quad AA = A^2 = \begin{bmatrix} 3 & 1 & -2 \\ -2 & 4 & 0 \\ 0 & 8 & -2 \end{bmatrix}.$$

Certain square matrices have special properties. For example, if the elements below the main diagonal are zero, the matrix is called an *upper*

triangular matrix. Thus,

$$U = \begin{bmatrix} 1 & -2 & -1 \\ 0 & 3 & 6 \\ 0 & 0 & 2 \end{bmatrix}$$

is an upper triangular. A square matrix, in which all elements above the main diagonal are zero, is called *lower triangular*. If only the elements in the main diagonal are nonzero, the matrix is called a *diagonal matrix*. An example of a diagonal matrix is

$$I_n = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

which is called the *identity matrix*.

EXERCISE SET 4.1

1. Given the matrices

$$A = \begin{bmatrix} 3 & 8 & 1 & -6 \\ 1 & 1 & 4 & 0 \\ 3 & 2 & 1 & -4 \end{bmatrix}, \quad B = \begin{bmatrix} -2 & 6 & 2 & -3 \\ 3 & 0 & 0 & -4 \\ -5 & 9 & 4 & -4 \end{bmatrix}, \quad C = \begin{bmatrix} -5 \\ -2 \\ 2 \\ 1 \end{bmatrix}.$$

   (a) Compute $A + B$, $-4A - 3B$.
   (b) Find AC, BC.
   (c) Find B^T, $(AB)^T$.

2. Given the matrices

$$A = \begin{bmatrix} -1 & 4 & 6 \\ 4 & -5 & 0 \\ 1 & 1 & -2 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 3 & 3 & 4 \\ -2 & 0 & -4 & 1 \\ 6 & 2 & 2 & -5 \end{bmatrix}$$

   (a) Compute AB, BA.
   (b) Find A^2, B^2.
   (c) $B^T A$.

4.2 NAIVE GAUSSIAN ELIMINATION

Consider the system (4.1) in matrix form

$$A\mathbf{x} = \mathbf{b}.$$

Let us denote the original system by $A^{(1)}\mathbf{x} = \mathbf{b}^{(1)}$. That is,

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \cdots & a_{2n}^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1}^{(1)} & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_n^{(1)} \end{bmatrix} \tag{4.2}$$

The Gaussian elimination consists of reducing the system (4.2) to an equivalent system $U\mathbf{x} = \mathbf{d}$, in which U is an upper triangular matrix. This new system can be easily solved by back substitution.

Algorithm: [.2cm]

Step 1: Assume $a_{11}^{(1)} \neq 0$. Define the row multipliers by

$$m_{i1} = \frac{a_{i1}^{(1)}}{a_{11}^{(1)}}$$

Multiply the first row by m_{i1} and subtract from the ith row ($i = 2, \ldots, n$) to get

$$\begin{aligned} a_{ij}^{(2)} &= a_{ij}^{(1)} - m_{i1}a_{1j}^{(1)}, \quad j = 2, 3, \cdots, n \\ b_i^{(2)} &= b_i^{(1)} - m_{i1}b_1^{(1)} \end{aligned}$$

Here, the first rows of A and $\mathbf{b}$ are left unchanged, and the entries of the first column of A below $a_{11}(1)$, are set to zeros.

The result of the transformed system is

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_n^{(2)} \end{bmatrix}$$

We continue in this way. At the kth step we have [.2cm]

Step k: Assume $a_{kk}^{(k)} \neq 0$. Define the row multipliers by

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

Multiply the kth row by m_{ik} and subtract from the ith row ($i = k+1, \ldots, n$) to get

$$\begin{aligned} a_{ij}^{(k+1)} &= a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}, \quad j = k+1, \ldots, n \\ b_i^{(k+1)} &= b_i^{(k)} - m_{ik} b_k^{(k)} \end{aligned}$$

At this step, the entries of column k below the diagonal element are set to zeros, and the rows 1 through k are left undisturbed. The result of the transformed system is

$$\begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & a_{1,k+1}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & a_{2,k+1}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{kk}^{(k)} & a_{k,k+1}^{(k)} & \cdots & a_{kn}^{(k)} \\ 0 & 0 & \cdots & 0 & a_{k+1,k+1}^{(k+1)} & \cdots & a_{k+1,n}^{(k+1)} \\ \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & a_{n,k+1}^{(k+1)} & \cdots & a_{nn}^{(k+1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1^{(1)} \\ b_2^{(2)} \\ \vdots \\ b_k^{(k)} \\ b_{k+1}^{(k+1)} \\ \vdots \\ b_n^{(k+1)} \end{bmatrix}$$

At $k = n-1$, we obtain the final triangular system

$$\begin{aligned} a_{11}^{(1)} x_1 + a_{12}^{(1)} x_2 + \quad \cdots \quad + a_{1n}^{(1)} x_n &= b_1^{(1)} \\ a_{22}^{(2)} x_2 + \quad \cdots \quad + a_{2n}^{(2)} x_n &= b_2^{(2)} \\ \cdots \quad &= \cdots \\ a_{n-1,n-1}^{(n-1)} x_{n-1} + a_{n-1,n}^{(n-1)} x_n &= b_{n-1}^{(n-1)} \\ a_{nn}^{(n)} x_n &= b_n^{(n)} \end{aligned}$$

Using back substitution, we obtain the following solution of the system

$$\begin{aligned} x_n &= \frac{b_n^{(n)}}{a_{nn}^{(n)}} \\ x_{n-1} &= \frac{b_{n-1}^{(n-1)} - a_{n-1,n}^{(n-1)} x_n}{a_{n-1,n-1}^{(n-1)}} \end{aligned}$$

$$x_i = \frac{b_i^{(i)} - (a_{i,i+1}^{(i)} x_{i+1} + \cdots + a_{in}^{(i)} x_n)}{a_{ii}^{(i)}}$$
$$= \frac{b_i^{(i)} - \sum_{j=i+1}^{n} a_{ij}^{(i)} x_j}{a_{ii}^{(i)}}, \quad i = n-2, n-3, \ldots, 1.$$

Remarks: In the Gaussian elimination algorithm described above, we used the equations in their natural order and we assumed at each step that the pivot element $a_{kk}^{(k)} \neq 0$. So the algorithm fails if the pivot element becomes zero during the elimination process. In order to avoid an accidental zero pivot, we use what is called Gaussian elimination with scaled partial pivoting. This method will be described in the next section.

Operation count. One way of measuring the efficiency of the algorithm of naive Gaussian elimination is to count the number of arithmetic operations required to obtain the solution of the system $A\mathbf{x} = \mathbf{b}$. The current convention is to count the number of multiplications and divisions only because most computers perform additions and subtractions much faster. Furthermore, the number of multiplications and divisions is counted together. Consider the naive Gaussian elimination algorithm described above. Step k requires

$$(n-k)^2 + (n-k) \quad \text{operations to find the new } a_{ij}$$

and

$$(n-k) \quad \text{operations to find the new } b_i$$

The operations must be done for $k = 1, 2, \cdots, n-1$. Hence, using the formulas

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}, \quad \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6},$$

the total number of operations is found to be

$$\frac{n^3 - n}{3} \quad \text{operations applied on the matrix } A$$

and

$$\frac{n^2 - n}{2} \quad \text{operations applied on the matrix } \mathbf{b}.$$

To solve the resulting triangular system, one needs $(n-i)$ multiplications and one division to get x_i. By summing this over $i = 1, 2, \ldots, n$, we get

$$\frac{n(n+1)}{2} \quad \text{operations to solve the triangular system.}$$

Thus, we have the following result:

LEMMA 4.1

The total number of multiplications and divisions required to obtain the solution of an $n \times n$ linear system using naive Gaussian elimination is

$$\frac{n^3}{3} + n^2 - \frac{n}{3}.$$

Hence, for n large the total number of operations is approximately $n^3/3$.

EXAMPLE 4.3

Solve the system of equation

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 5 \\ -1 & 1 & -5 & 3 \\ 3 & 1 & 7 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 31 \\ -2 \\ 18 \end{bmatrix}$$

The augmented matrix along with the row multipliers m_{i1}, are

$$\begin{array}{r} \text{pivotal element} \rightarrow \\ m_{21} = 2 \\ m_{31} = -1 \\ m_{41} = 3 \end{array} \left[\begin{array}{rrrr|r} \mathbf{1} & 1 & 1 & 1 & 10 \\ 2 & 3 & 1 & 5 & 31 \\ -1 & 1 & -5 & 3 & -2 \\ 3 & 1 & 7 & -2 & 18 \end{array}\right]$$

Subtracting multiples of the first equation from the three others, gives

$$\begin{array}{r} \\ \text{pivotal element} \rightarrow \\ m_{32} = 2 \\ m_{42} = -2 \end{array} \left[\begin{array}{rrrr|r} 1 & 1 & 1 & 1 & 10 \\ 0 & \mathbf{1} & -1 & 3 & 11 \\ 0 & 2 & -4 & 4 & 8 \\ 0 & -2 & 4 & -5 & -12 \end{array}\right]$$

m-function 4.1

The following MATLAB function **ngaussel.m** finds the solution of a linear system using naive Gaussian elimination. INPUTS are an *nxn* matrix *A* and an *n x 1* coefficient vector *b*.

```
function ngaussel(A,b)
% Solve the system Ax=b using naive gaussian elimination
n=length(b);
x=zeros(n,1);
fprintf('\n');
disp('     The augmented matrix is')
augm =[A b]
for k=1:n-1
  for i=k+1:n
    m=A(i,k)/A(k,k);
    for j=k+1:n
      A(i,j)=A(i,j)-m*A(k,j);
    end
    A(i,k)=m;
    b(i)=b(i)-m*b(k);
  end
end
x(n)=b(n)/A(n,n);
for i=n-1:-1:1
  S=b(i);
  for j=i+1:n
    S=S-A(i,j)*x(j);
  end
  x(i)=S/A(i,i);
end
% Print the results
fprintf('\n');
disp(' The transformed upper triangular augmented matrix C is =')
fprintf('\n');
for i=1:n
  for j=1:n
    if (j<i) A(i,j)=0; end
  end
end
C=[A b]
fprintf('\n');
disp('     The vector solution is =')
x
```

```
» A=[1 1 1 1;2 3 1 5;-1 1 -5 3;3 1 7 -2];
» b=[10 31 -2 18]';
» ngaussel(A,b)

   The augmented matrix is

augm =

   1   1   1   1  10
   2   3   1   5  31
  -1   1  -5   3  -2
   3   1   7  -2  18

The transformed upper triangular augmented matrix C is =

C =
   1   1   1   1  10
   0   1  -1   3  11
   0   0  -2  -2 -14
   0   0   0  -1  -4

   The vector solution is =

x =
        1
        2
        3
        4
```

Table 4.1 Naive Gaussian elimination for Example 4.3.

Subtracting multiples, of the second equation from the last two, gives

$$\begin{array}{r} \\ \\ \text{pivotal element} \rightarrow \\ m_{43} = -1 \end{array} \left[\begin{array}{rrrr|r} 1 & 1 & 1 & 1 & 10 \\ 0 & 1 & -1 & 3 & 11 \\ 0 & 0 & \mathbf{-2} & -2 & -14 \\ 0 & 0 & 2 & 1 & 10 \end{array}\right]$$

Subtracting multiples of the third equation from the last one, gives the upper triangular system

$$\left[\begin{array}{rrrr|r} 1 & 1 & 1 & 1 & 10 \\ 0 & 1 & -1 & 3 & 11 \\ 0 & 0 & -2 & -2 & -14 \\ 0 & 0 & 0 & -1 & -4 \end{array}\right]$$

The process of the back substitution algorithm applied to the triangular system produces the solution

$$
\begin{aligned}
x_4 &= \frac{-4}{-1} = 4 \\
x_3 &= \frac{-14 + 2x_4}{-2} = \frac{-6}{-2} = 3 \\
x_2 &= 11 + x_3 - 3x_4 = 11 + 3 - 12 = 2 \\
x_1 &= 10 - x_2 - x_3 - x_4 = 10 - 2 - 3 - 4 = 1
\end{aligned}
$$

given in Table 4.1.

EXAMPLE 4.4

Solve the linear system of equations

$$
\begin{bmatrix}
-1 & 1 & -2 & 0 & 4 & -2 & 2 \\
1 & 4 & -2 & -4 & -6 & -2 & 2 \\
-1 & 1 & 6 & 2 & 8 & 2 & 8 \\
1 & -1 & -1 & -2 & 6 & -4 & -8 \\
-1 & 1 & 1 & 2 & 4 & -4 & 2 \\
1 & -1 & -1 & 1 & 1 & 4 & 4 \\
-1 & 1 & 1 & -1 & -1 & 1 & 4
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}
=
\begin{bmatrix} 2 \\ 4 \\ 6 \\ 8 \\ -2 \\ -4 \\ -6 \end{bmatrix}
$$

The computed results are shown in Table 4.2.

EXERCISE SET 4.2

1. Write the following systems of equations in matrix form and then solve them:

(a)
$$
\begin{aligned}
2x + y + z &= 7 \\
2x + 2y + 3z &= 10 \\
-4x + 4y + 5z &= 14
\end{aligned}
$$

```
» A=[-1 1 -2 0 4 -2 2
  1 4 -2 -4 -6 -2 2
 -1 1 6 2 8 2 8
  1 -1 -1 -2 6 -4 -8
 -1 1 1 2 4 -4 2
  1 -1 -1 1 1 4 4
 -1 1 1 -1 -1 1 4];
» b=[2 4 6 8 -2 -4 -6]';
» gaussel(A,b)

   The augmented matrix is

augm =

  -1   1  -2   0   4  -2   2   2
   1   4  -2  -4  -6  -2   2   4
  -1   1   6   2   8   2   8   6
   1  -1  -1  -2   6  -4  -8   8
  -1   1   1   2   4  -4   2  -2
   1  -1  -1   1   1   4   4  -4
  -1   1   1  -1  -1   1   4  -6

 The transformed upper triangular augmented matrix C is =

C =
        -1         1        -2         0         4        -2         2         2
         0         5        -4        -4        -2        -4         4         6
         0         0         8         2         4         4         6         4
         0         0         0     -1.25      11.5      -4.5     -3.75      11.5
         0         0         0         0        10        -8        -6         6
         0         0         0         0         0     15.28     16.56      2.04
         0         0         0         0         0         0      2.58    -10.67

   The vector solution is =

x =

   -0.2215
   11.1801
   -0.4239
    3.2485
    1.8103
    4.6126
   -4.1329
```

Table 4.2 Naive Gaussian elimination for Example 4.4.

(b)
$$\begin{aligned} x + y + 7z + v + 2w &= 1 \\ 2x + 1y + 6z + 6v + 3w &= -2 \\ - y + 2z + 3v + 2w &= -3 \\ 5x - y + 4z + 7v + w &= 3 \\ 2x + 3y + 2z + w &= 4 \end{aligned}$$

(c)
$$\begin{aligned} 3x + 5y - 2z - w &= -12 \\ -2x + y - 9z - 2w &= 1 \\ x + 7y - 6z - 3w &= 10 \\ -3x + y - z - 4w &= 7 \end{aligned}$$

2. Solve the following tridiagonal system of 10 equations by 'hand' or with the computer.

$$\begin{bmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & 0 \\ & & \ddots & & \\ 0 & & 1 & -4 & 1 \\ & & & 1 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} 0.5 \\ 2 \\ \vdots \\ 2 \\ 0.5 \end{bmatrix}$$

3. Consider the system

$$\begin{bmatrix} 3 \cdot 10^{-n} & 3.0 \\ 1.0 & 1.0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2.0 + 10^{-n} \\ 1.0 \end{bmatrix}$$

Its exact solution is $x_1 = 1/3$, $x_2 = 2/3$.

(a) Compute the numerical solution for $n = 4, 8, 12$.

(b) Compute the residual vector $\bar{\mathbf{r}} = A\bar{\mathbf{x}} - \mathbf{b}$, and the error vector $\bar{\mathbf{e}} = \mathbf{x} - \bar{\mathbf{x}}$, for the different values of n given in (a). $\bar{\mathbf{x}}$ denotes the computed solution and $\mathbf{x}$ the exact solution.

4. Consider the system

$$\begin{bmatrix} 1 & 1 & 7 & 8 \\ -3 & -3 & -1 & 6 \\ 0 & 2 & 2 & 7 \\ 5 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Attempt to solve the system by Gaussian elimination. Explain what happens.

5. The Hilbert matrix A of order n is defined by

$$a_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, 2, \ldots, n$$

   (a) Write down the matrix A for $n = 5$.

   (b) If $\mathbf{b} = [1, 0, 0, 0, 0]^T$, solve the system $\mathbf{Ax} = \mathbf{b}$ using Gaussian elimination.

6. **Computer assignment:**
   Write a computer program in FORTRAN or C to solve a system of n linear equations and n unknowns using naive Gaussian elimination.

   Input data to the program should be:

   (a) The number of equations n.

   (b) The augmented matrix.

   Output should consist of:

   (a) The augmented matrix.

   (b) The vector solution.

   Test your program to solve Exercise 1(b).

4.3 GAUSSIAN ELIMINATION WITH SCALED PARTIAL PIVOTING

Having a zero pivot element is not the only source of trouble that can arise when we apply naive Gaussian elimination. Under certain circumstances, the pivot element can become very small in magnitude compared to the rest of the elements in the pivot row. This can dramatically increase the round-off error, which can result in an inaccurate solution vector. To illustrate some of the effects of round-off error in the elimination process, we apply naive Gaussian elimination to the system

$$0.0002x_1 + 1.471x_2 = 1.473$$
$$0.2346x_1 - 1.317x_2 = 1.029$$

using four-digit floating-point arithmetic with rounding. The exact solution of this system is $x_1 = 10.00$ and $x_2 = 1.000$.
The multiplier for this system is

$$m_{21} = \frac{0.2346}{0.0002} = 1173.$$

Applying naive Gaussian elimination and performing the appropriate rounding, gives

$$\begin{aligned} 0.0002x_1 + 1.471x_2 &= \ \ 1.473 \\ -\ 1726.x_2 &= -1727 \end{aligned}$$

Hence,

$$\begin{aligned} x_2 &= \frac{-1727.}{-1726.} \\ &= 1.001 \\ x_1 &= \frac{1.473 - (1.471)(1.001)}{0.0002} \\ &= \frac{1.473 - 1.472}{0.0002} \\ &= 5.000 \end{aligned}$$

As one can see, x_2 is a close approximation of the actual value. However, the relative error in the computed solution for x_1 is very large: 50%. The failure of naive Gaussian elimination in this example, results from the fact that $|a_{11}| = 0.0002$ is small compared to $|a_{12}|$. Hence, a relatively small error due to round-off in the computed value, x_2, led to a relatively large error in the computed solution, x_1.

A useful strategy to avoid the problem of having a zero or very small pivot element, is to use Gaussian elimination with scaled partial pivoting. In this method, the equations of the system (4.2) are used in a different order, and the pivot equation is selected by looking for the absolute largest coefficient of x_k relative to the size of the equation. We shall illustrate the method by solving the system

$$\begin{bmatrix} 1 & 3 & -2 & 4 \\ 2 & -3 & 3 & -1 \\ -1 & 7 & -4 & 2 \\ 3 & -1 & 6 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -11 \\ 6 \\ -9 \\ 15 \end{bmatrix}. \tag{4.3}$$

At the beginning, the index vector is $\mathbf{d} = [1, 2, 3, 4]$. We first define the scaled vector by

$$\mathbf{c} = [c_1, c_2, c_3, c_4]$$

where

$$c_i = \max_{1 \le j \le 4} |a_{ij}|, \quad i = 1, 2, 3, 4.$$

For our example (4.3) the scaled vector is

$$\mathbf{c} = [4, 3, 7, 6]$$

we emphasize that the scaled vector will remain unchanged through the entire elimination process.

In order to determine the index vector, we now define the ratio vector by

$$\mathbf{r} = [r_1, r_2, r_3, r_4]$$

where

$$r_i = \frac{|a_{i1}|}{c_i}, \quad i = 1, 2, 3, 4.$$

In our example, it is

$$\mathbf{r} = \left[\frac{1}{4}, \frac{2}{3}, \frac{1}{7}, \frac{1}{2}\right]$$

since 2/3 is the largest element in $\mathbf{r}$, we interchange elements 1 and 2 in the index vector to get

Step 1: $\mathbf{d} = [2, 1, 3, 4]$. This means that row 2 is the pivot equation. We now use Gaussian elimination to get

$$\begin{bmatrix} 0 & 9/2 & -7/2 & 9/2 \\ 2 & -3 & 3 & -1 \\ 0 & 11/2 & -5/2 & 3/2 \\ 0 & 7/2 & 3/2 & 7/2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -14 \\ 6 \\ -6 \\ 6 \end{bmatrix}$$

We continue in this way.

Step 2: $\mathbf{r} = \left[\frac{9}{8}, \frac{11}{14}, \frac{7}{12}\right]$

As 9/8 is the largest element in $\mathbf{r}$, nothing is to be interchanged in the index vector, thus

$$\mathbf{d} = [2, 1, 3, 4]$$

This means that row 1 becomes the new pivot equation.
We now apply Gaussian elimination again to get

$$\begin{bmatrix} 0 & 9/2 & -7/2 & 9/2 \\ 2 & -3 & 3 & -1 \\ 0 & 0 & 16/9 & -4 \\ 0 & 0 & 38/9 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -14 \\ 6 \\ 100/9 \\ 152/9 \end{bmatrix}$$

Step 3: $\mathbf{r} = [\frac{16}{63}, \frac{38}{54}]$

As $38/9 > 16/9$ in $\mathbf{r}$ we interchange row 3 and 4 to get the new index vector

$$\mathbf{d} = [2, 1, 4, 3]$$

This means that row 4 becomes the new pivot row. We now apply Gaussian elimination to get

$$\begin{bmatrix} 0 & 9/2 & -7/2 & 9/2 \\ 2 & -3 & 3 & -1 \\ 0 & 0 & 0 & -4 \\ 0 & 0 & 38/9 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -14 \\ 6 \\ 4 \\ 152/9 \end{bmatrix}$$

Finally, back substitution produces the solution

$$\begin{aligned} x_4 &= \frac{4}{-4} = -1 \\ x_3 &= \frac{1}{38/9}(152/9) = 4 \\ x_2 &= \frac{1}{9/2}[-14 + 4(7/2) - 1(-9/2)] = 1 \\ x_1 &= \frac{1}{1/2}[6 + 3 - 12 - 1] = -2 \end{aligned}$$

Note that during the elimination process, only elements in the index vector $\mathbf{d}$ have been interchanged and not the equations.

EXAMPLE 4.5

Solve the system of equation using Gaussian elimination with scaled partial pivoting

$$\begin{bmatrix} 1 & 3 & -2 & 4 \\ 2 & -3 & 3 & -1 \\ -1 & 7 & -4 & 2 \\ 3 & -1 & 6 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -11 \\ 6 \\ -9 \\ 15 \end{bmatrix}$$

The computed results are shown in Table 4.3.

m-function 4.2

The following MATLAB function **gaussel.m** finds the solution of a linear system using Gaussian elimination with scaled partial pivoting. INPUTS are an *n x n* matrix *A* and an *n x 1* coefficient vector *b*.

```
function gaussel(A,b)
% Solve the system Ax=b using Gaussian elimination with scaled
% partial pivoting.
n=length(b);
x=zeros(n,1);
fprintf('\n');
disp('     The augmented matrix is =')
augm =[A b]
for i=1:n
  d(i)=i;
  smax=0;
  for j=1:n
    smax=max(smax,abs(A(i,j)));
  end
  c(i)=smax;
end
for k=1:n-1
  rmax=0;
  for i=k:n
    R=abs(A(d(i),k))/c(d(i));
    if (R>rmax)
      j=i;
      rmax=R;
    end
  end
  dk=d(j);
  d(j)=d(k);
  d(k)=dk;
  for i=k+1:n
    m=A(d(i),k)/A(dk,k);
    for j=k+1:n
      A(d(i),j)=A(d(i),j)-m*A(dk,j);
    end
    A(d(i),k)=m;
  end
end
% Perform the back substitution.
```

```
for k=1:n-1
  for i=k+1:n
     b(d(i))=b(d(i))-b(d(k))*A(d(i),k);
   end
end
x(n)=b(d(n))/A(d(n),n);
for i=n-1:-1:1
  S=b(d(i));
  for j=i+1:n
    S=S-A(d(i),j)*x(j);
  end
  x(i)=S/A(d(i),i);
end
disp('The scale vector =')
c
disp('The index vector =')
d
fprintf('\n');
disp(' The transformed upper triangular augmented matrix C is =')
fprintf('\n');
for i=1:n
  M(i,:)=A(d(i),:);
end
for i=1:n
  for j=1:n
    if (j<i) M(i,j)=0; end
  end
end
C=[M b]
fprintf('\n');
disp('     The vector solution is =')
x
```

EXAMPLE 4.6

Solve the system of equation using Gaussian elimination with scaled partial pivoting

$$\begin{bmatrix} 0.0002 & 1.471 \\ 0.2346 & -1.317 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.473 \\ 1.029 \end{bmatrix}$$

The computed results are shown in Table 4.4.

```
» A=[1 3 -2 4;2 -3 3 -1;-1 7 -4 2;3 -1 6 2];
» b=[-11 6 -9 15]';
» gaussel(A,b)

    The augmented matrix is =

 augm =

    1    3   -2    4   -11
    2   -3    3   -1     6
   -1    7   -4    2    -9
    3   -1    6    2    15

The scale vector =

 c =
      4    3    7    6

The index vector =

d =
       2    1    4    3

 The transformed upper triangular augmented matrix C is =

C =
            0        4.5       -3.5        4.5       -14
            2         -3          3         -1         6
            0          0          0         -4         4
            0          0       4.22          0     16.89

    The vector solution is =

 x =
          -2
           1
           4
          -1
```

Table 4.3 Gaussian elimination for Example 4.5.

```
» A=[0.0002  1.471
     0.2346 -1.317];
» b=[1.473 1.029]';
» gaussel(A,b)

   The augmented matrix is

augm =

  0.0002   1.4710   1.4730
  0.2346  -1.3170   1.0290

The scale vector =

c =
     1.4710   1.3170

The index vector =

d =
     2    1

The transformed upper triangular augmented matrix C is

C =
                 0      1.4721     1.4721
            0.2346      -1.317      1.029

   The vector solution is

x =
        10
         1
```

Table 4.4 Gaussian elimination for Example 4.6.

EXERCISE SET 4.3

1. Solve the following systems using Gaussian elimination with partial scaled pivoting:

(a)
$$\begin{aligned} 6x + 4y + 13z &= -23 \\ 2x + y - z &= 4 \\ -3x + 6y - z &= 8 \end{aligned}$$

(b) $$\begin{bmatrix} -2 & -3 & 1 & 2 \\ 7 & 6 & 0 & -3 \\ 0 & 3 & 1 & 5 \\ 2 & -2 & 6 & 6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ -4 \\ 1 \\ 8 \end{bmatrix}$$

2. Consider the system

$$\begin{bmatrix} 3 & 3 & 7 \\ 1 & 1 & -1 \\ -2 & -2 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 4 \\ 3 \end{bmatrix}$$

Attempt to solve the system by Gaussian elimination with scaled partial pivoting. Explain what happens.

3. If the Gaussian algorithm with scaled partial pivoting is used on the following system, what is the scaled vector? What is the second pivot row?

$$\begin{bmatrix} 2 & 4 & 29 \\ -1 & 5 & 4 \\ 1 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 15 \\ -3 \\ 1 \end{bmatrix}$$

4. Solve the system

$$\begin{bmatrix} -1 & 2 & -1 & 0 & 4 \\ 1 & 2 & 0 & 3 & 0 \\ 0 & -3 & 1 & 1 & 2 \\ 1 & 0 & 2 & -1 & 3 \\ 2 & -2 & 2 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 0 \\ 3 \end{bmatrix}$$

using Gaussian elimination with scaled partial pivoting. Find the scaled and index vectors at each step.

5. Compute the inverse A^{-1}, where

$$A = \begin{bmatrix} 2 & 1 & 2 \\ 1 & 2 & 3 \\ 4 & 1 & 2 \end{bmatrix}$$

by solving the system $AX = I$, using scaled partial pivoting.

6. Find the scale vector and the third pivot row if Gaussian elimination with scaled partial pivoting is used on the matrix

$$\begin{bmatrix} -2 & 4 & 5 & -1 \\ -1 & 2 & 9 & 4 \\ 6 & 2 & -9 & -5 \\ 3 & 0 & 3 & -8 \end{bmatrix}$$

7. Use both naive Gaussian elimination and Gaussian elimination with scaled partial pivoting to solve the following linear system using four-decimal floating point arithmetic

$$\begin{aligned} 0.0003x_1 + 1.354x_2 &= 1.357 \\ 0.2322x_1 - 1.544x_2 &= 0.7780. \end{aligned}$$

8. Use Gaussian elimination with scaled partial pivoting and two-digit rounding arithmetic to solve the linear system

$$\begin{aligned} 0.0001x_1 + 66.x_2 &= 65. \\ 2.0x_1 - 28.x_2 &= 33. \end{aligned}$$

9. **Computer assignment:**
Write a computer program in FORTRAN or C to solve a system of n linear equations and n unknowns using Gaussian elimination with scaled partial pivoting.
Input data to the program should be

   (a) The number of equations n.

   (b) The augmented matrix.

   Output should consist of:

   (a) The augmented matrix.

   (b) The vector solution.

   Test your program to solve Exercise 1(a).

4.4 LU DECOMPOSITION

Consider the system of equations

$$A\mathbf{x} = \mathbf{b}$$

The LU decomposition consists of transforming the coefficient matrix A into the product of two matrices, L and U, where L is a lower triangular matrix and U is an upper triangular matrix having 1's on its diagonal.

Once L and U are found, the solution of the system $A\mathbf{x} = \mathbf{b}$ can be carried out by writing

$$LU\mathbf{x} = \mathbf{b}$$

and setting

$$U\mathbf{x} = \mathbf{y} \tag{4.4}$$

so that

$$L\mathbf{y} = \mathbf{b} \tag{4.5}$$

Eqns (4.4) and (4.5) are two triangular systems which can be easily solved by first using the forward substitution in (4.5) to get $\mathbf{y}$, and then with $\mathbf{y}$ known, we use the back substitution in (4.4) to get $\mathbf{x}$. Two types of factorizations will now be presented, the first one uses Crout's and Choleski's methods and the second one uses the Gaussian elimination method.

4.4.1 Crout's and Choleski's methods

We shall illustrate the method of finding L and U in the case of a 4×4 matrix:
We wish to find L, having nonzero diagonal entries, and U such that

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & {}_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{1n} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

Multiplying the rows of L by the first column of U, one gets

$$l_{i1} = a_{i1}, \quad i = 1, 2, 3, 4.$$

Hence, the first column of L is given by the first column of A. Next, multiply the columns of U by the first row of L to get

$$l_{11}u_{1i} = a_{1i}, \quad i = 2, 3, 4.$$

Thus,

$$u_{1i} = \frac{a_{1i}}{l_{11}}, \quad i = 2, 3, 4.$$

which give the first row of U.

We continue in this way by getting alternatively a column of L and a row of U. The result is

$$\begin{aligned}
l_{i2} &= a_{i2} - l_{i1}u_{12}, && i = 2, 3, 4.\\
u_{2i} &= \frac{a_{2i} - l_{21}u_{1i}}{l_{22}}, && i = 3, 4.\\
l_{i3} &= a_{i3} - l_{i1}u_{13} - l_{i2}u_{23}, && i = 3, 4.\\
u_{34} &= \frac{a_{34} - l_{31}u_{14} - l_{32}u_{24}}{l_{33}},\\
l_{44} &= a_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34}.
\end{aligned}$$

In algorithmic form, the factorization may be presented as follows for an $n \times n$ matrix:

$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik}u_{kj}, \quad j \le i,\ i = 1, 2, \ldots, n. \tag{4.6}$$

$$u_{ij} = \frac{a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}}{l_{ii}}, \quad i \le j,\ j = 2, 3, \ldots, n. \tag{4.7}$$

Note that this algorithm can be applied if the diagonal elements l_{ii}, for each $i = 1, \ldots, n$, of L, are nonzero.

The LU factorization that we have just described, requiring the diagonal elements of U to be one, is known as **Crout's method**. If instead the diagonal of L is required to be one, the factorization is called **Doolittle's method**.

In the case when the $n \times n$ matrix A is symmetric ($A = A^T$), and positive definite, that is

$$\mathbf{x}^T A\mathbf{x} > 0, \quad \text{for all nonzero } n\text{-vectors } \mathbf{x}$$

then it is possible to carry out a factorization without any need for pivoting or scaling. This factorization is known as **Cholesky's method**, and A can be factored in the form

$$A = LL^T$$

where L is a lower triangular matrix. The construction of L is similar to the one used for Crout's method. Multiplying L by L^T and setting the result equal to A gives

$$l_{ii} = \left[a_{ii} - \sum_{k=1}^{i-1}(l_{ik})^2\right]^{1/2}, \qquad i = 1, 2, \ldots, n$$

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}}{l_{jj}}, \qquad i = j+1, j+2, \ldots, n;\ j = 1, 2, \ldots, n. \tag{4.8}$$

EXAMPLE 4.7

Factor the following matrix using Cholesky's method.

$$A = \begin{bmatrix} 16 & 4 & 4 \\ 4 & 26 & 6 \\ 4 & 6 & 11 \end{bmatrix}$$

From (4.8) we have

$l_{11} = \sqrt{16} = 4$

$l_{21} = 4/4 = 1, \quad l_{22} = \sqrt{26-1} = 5$

$l_{31} = 4/4 = 1, \quad l_{32} = (6-1)/5 = 1, \quad l_{33} = \sqrt{11-1-1} = 3$

Thus,

$$L = \begin{bmatrix} 4 & 0 & 0 \\ 1 & 5 & 0 \\ 1 & 1 & 3 \end{bmatrix} \quad \text{and} \quad U = L^T \begin{bmatrix} 4 & 1 & 1 \\ 0 & 5 & 1 \\ 0 & 0 & 3 \end{bmatrix}.$$

The LU method is particularly useful when it is necessary to solve a whole series of systems

$$A\mathbf{x} = \mathbf{b}_1,\ A\mathbf{x} = \mathbf{b}_2,\ \ldots\ A\mathbf{x} = \mathbf{b}_n$$

each of which has the same square coefficient matrix A.

Many problems in science and engineering involve systems of this type, and in this case it is more efficient to use the LU method than separately applying the Gaussian elimination method to each of the k

systems. Storage space may be economized since there is no need to store the zeros in either L or U, and the ones on the diagonal of U. Eqn. (4.8) shows that once any element of A, a_{ij}, is used, it never again appears in the equations and its place can be used to store element, of either L or U. In other words, the L and U matrices are constructed by storing theirs elements in the space of A.

EXAMPLE 4.8

Use Crout's method to solve the system

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 5 \\ -1 & 1 & -5 & 3 \\ 3 & 1 & 7 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 10 \\ 31 \\ -2 \\ 18. \end{bmatrix}$$

If A has a direct factorization LU, then

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 3 & 1 & 5 \\ -1 & 1 & -5 & 3 \\ 3 & 1 & 7 & -2 \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} 1 & u_{12} & u_{13} & u_{14} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By multiplying L with U and comparing the elements of the product matrix with those of A, we obtain:

(i) Multiplication of the first row of L with the columns of U gives

$$\begin{aligned} &l_{11} = 1, \\ &l_{11}u_{12} = 1 \Longrightarrow u_{12} = 1, \\ &l_{11}u_{13} = 1 \Longrightarrow u_{13} = 1, \\ &l_{11}u_{14} = 1 \Longrightarrow u_{14} = 1. \end{aligned}$$

(ii) Multiplication of the second row of L with the columns of U gives

$$\begin{aligned} &l_{21} = 2, \\ &l_{21}u_{12} + l_{22} = 3 \Longrightarrow l_{22} = 3 - l_{21}u_{12} = 1, \\ &l_{21}u_{13} + l_{22}u_{23} = 1 \Longrightarrow u_{23} = (1 - l_{21}u_{13})/l_{22} = -1, \\ &l_{21}u_{14} + l_{22}u_{24} = 5 \Longrightarrow u_{24} = (5 - l_{21}u_{14})/l_{22} = 3. \end{aligned}$$

m-function 4.3

The following MATLAB function **lufact.m** finds the solution of a linear system using Crout's LU decomposition. INPUTS are an *n x n* matrix *A* and an *n x 1* coefficient vector *b*.

```
function lufact(A,b)
% Solve the system Ax=b using the LU decomposition.
n=length(b);
y=zeros(n,1);
x=zeros(n,1);
fprintf('\n');
for i=1:n
  U(i,i)=1;
end
L(1,1)=A(1,1)/U(1,1);
for j=2:n
  L(j,1)=A(j,1)/U(1,1);
  U(1,j)=A(1,j)/L(1,1);
end
for i=2:n-1
  S=0;
  for k=1:i-1
    S=S+U(k,i)*L(i,k);
  end
  L(i,i)=(A(i,i)-S)/U(i,i);
  for j=i+1:n
    S=0;
    for k=1:i-1
      S=S+U(k,i)*L(j,k);
    end
    L(j,i)=(A(j,i)-S)/U(i,i);
    S=0;
    for k=1:i-1
      S=S+U(k,j)*L(i,k);
    end
    U(i,j)=(A(i,j)-S)/L(i,i);
  end
end
S=0;
for k=1:n-1
  S=S+U(k,n)*L(n,k);
end
L(n,n)=(A(n,n)-S)/U(n,n);
```

```
% Perform the forward substitution.
y(1)=b(1)/L(1,1);
for i=2:n
  S=b(i);
  for j=1:i-1
    S=S-L(i,j)*y(j);
  end
  y(i)=S/L(i,i);
end
% Perform the back substitution.
x(n)=y(n)/U(n,n);
for i=n-1:-1:1
  S=y(i);
  for j=i+1:n
    S=S-U(i,j)*x(j);
  end
  x(i)=S/U(i,i);
end
L
disp('      The forward substitution gives')
y
U
disp('      The vector solution is =')
x
```

(iii) Multiplication of the third row of L with the columns of U gives

$$
\begin{aligned}
&l_{31} = -1,\\
&l_{31}u_{12} + l_{32} = 1 \Longrightarrow l_{32} = 1 - l_{31}u_{12} = 2,\\
&l_{31}u_{13} + l_{32}u_{23} + l_{33} = -5 \Longrightarrow l_{33} = -5 - l_{31}u_{13} - l_{32}u_{23} = -2,\\
&l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} = 3 \Longrightarrow u_{34} = (3 - l_{31}u_{14} - l_{32}u_{24})/l_{33} = 1.
\end{aligned}
$$

(iv) Multiplication of the fourth row of L with the columns of U gives

$$
\begin{aligned}
&l_{41} = 3,\\
&l_{41}u_{12} + l_{42} = 1 \Longrightarrow l_{42} = 1 - l_{41}u_{12} = -2,\\
&l_{41}u_{13} + l_{42}u_{23} + l_{43} = 7 \Longrightarrow l_{43} = 7 - l_{41}u_{13} - l_{42}u_{23} = 2,\\
&l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44} = -2 \Longrightarrow\\
&l_{44} = -2 - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34} = -1.
\end{aligned}
$$

Hence,

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ -1 & 2 & -2 & 0 \\ 3 & -2 & 2 & -1 \end{bmatrix} \quad \text{and} \quad U = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & -1 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

By applying the forward substitution to the lower triangular system $L\mathbf{y} = \mathbf{b}$, we get

$$\begin{aligned} y_1 &= 10 \\ y_2 &= 31 - 2(10) = 11 \\ y_3 &= [-2 + 10 - 2(11)]/(-2) = 7 \\ y_4 &= -[18 - 3(10) + 2(11) - 2(7)] = 4 \end{aligned}$$

Finally, by applying the back substitution to the upper triangular system $U\mathbf{x} = \mathbf{y}$, we get

$$\begin{aligned} x_1 &= 10 - 4 - 3 - 2 = 1 \\ x_2 &= -[11 - 4 - 3(3)] = 2 \\ x_3 &= 7 - 4 = 3 \\ x_4 &= 4. \end{aligned}$$

The computed results are shown in Table 4.5.

4.4.2 Gaussian elimination method

We shall now illustrate a method of constructing L and U using Gaussian elimination. We process exactly as Gaussian elimination except that we keep a record of the elementary row operation performed at each step.

We concentrate here on the factorization without pivoting. So, we assume that the naive Gaussian elimination can be successfully performed to solve the linear system $A\mathbf{x} = b$. Also, we are not concerned with the coefficient vector b. That is, we do not need to form an augmented matrix. Looking back at the elimination process described in Section 4.2, we see that the row multipliers for naive Gaussian elimination at kth step are defined by

$$m_{ik} = \frac{a_{ik}^{(k)}}{a_{kk}^{(k)}}$$

```
» A=[1 1 1 1;2 3 1 5;-1 1 -5 3;3 1 7 -2];
» b=[10 31 -2 18]';
» lu(A,b)

L =

     1     0     0     0
     2     1     0     0
    -1     2    -2     0
     3    -2     2    -1

   The forward substitution gives
y =
    10
    11
     7
     4

U =

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

   The vector solution is =
x =
     1
     2
     3
     4
```

Table 4.5 Crout's method for Example 4.8.

provided that $a_{kk}^{(k)} \neq 0$. It is easily verified that when the matrix

$$M_1 = \begin{bmatrix} 1 & & & \\ -m_{21} & 1 & & 0 \\ -m_{31} & 0 & 1 & \\ \vdots & \vdots & & \ddots \\ -m_{n1} & 0 & \cdots & 0 \quad 1 \end{bmatrix}$$

is multiplied by the matrix A on the left, the result is

$$M_1 A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \cdots & a_{nn}^{(2)} \end{bmatrix}$$

which has the same effect of multiplying row 1 of A by the row multiplier m_{i1} and subtracting the result from row i of A for $i = 2, ..., n$. If $a_{kk}^{(k-1)} \neq 0$, the $k-1$ step is formed by

$$M_{k-1} M_{k-2} \cdots M_1 A = \begin{bmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \cdots & a_{1k}^{(1)} & \cdots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \cdots & a_{2k}^{(2)} & \cdots & a_{2n}^{(2)} \\ & 0 & \ddots & \vdots & & \vdots \\ & & & a_{kk}^{(k)} & \cdots & a_{kn}^{(k)} \\ \vdots & \vdots & & \vdots & & \vdots \\ 0 & 0 & \cdots & a_{nk}^{(k)} & \cdots & a_{nn}^{(k)} \end{bmatrix}$$

where

$$M_{k-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & 0 \\ & & -m_{k,k-1} & \ddots & \\ & & \vdots & & \\ & & -m_{n,k-1} & & 1 \end{bmatrix}.$$

After $(n-1)$ steps, elimination process without pivoting results in

$$M_{n-1} M_{n-2} \cdots M_1 A = \begin{bmatrix} a_{11}^{(1)} & \cdots & & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & & a_{2n}^{(2)} \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & a_{nn}^{(n)} \end{bmatrix} = U \tag{4.9}$$

which is an upper triangular matrix. It easy to verify directly that $M_i^{-1} M_i = I$, $i = 1, ..., n-1$. Using this result, we multiply Eqn. (4.9) on the left by M_{n-1}^{-1}, then $M_{n-2}^{-1}, ...$, to get

$$A = M_1^{-1} M_2^{-1} \cdots M_{n-1}^{-1} U.$$

It is also easy to verify directly that

$$M_{k-1}^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & 0 \\ & & m_{k,k-1} & \ddots & \\ & & \vdots & & \\ & & m_{n,k-1} & & 1 \end{bmatrix}.$$

Therefore,

$$M_1^{-1}M_2^{-1}\cdots M_{n-1}^{-1} = \begin{bmatrix} 1 & & & \\ m_{21} & 1 & & 0 \\ m_{31} & m_{32} & \ddots & \\ \vdots & \vdots & & 1 \\ m_{n1} & m_{n2} & \cdots & m_{n,n-1} \; 1 \end{bmatrix} = L \tag{4.10}$$

which is a lower triangular matrix with all ones in its diagonal. Finally, we see that

$$LU = M_1^{-1}M_2^{-1}\cdots M_{n-1}^{-1}M_{n-1}M_{n-2}\cdots M_1 A = A$$

where U is given by Eqn. (4.9) and L given by Eqn. (4.10).

EXAMPLE 4.9

Use Gaussian elimination to construct the LU decomposition of the matrix

$$A = \begin{bmatrix} 1 & 1 & 6 \\ -1 & 2 & 9 \\ 1 & -2 & 3 \end{bmatrix}$$

The matrix L will be constructed from the identity matrix placed at the left of A. We start with

$$A = I\,A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 6 \\ -1 & 2 & 9 \\ 1 & -2 & 3 \end{bmatrix}$$

The row multipliers are $m_{21} = -1$ and $m_{31} = 1$. Subtract from row 2 the multiple m_{21} of row 1 and from row 3 the multiples m_{21} of row 1, to get

$$A = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 6 \\ 0 & 3 & 15 \\ 0 & -3 & -3 \end{bmatrix}.$$

Note that the multipliers m_{21} and m_{21} are put below the diagonal element in the first column of the left matrix. The row multiplier of the second row is $m_{32} = -1$. Subtract from row 3 the multiple m_{32} of row 2 to get the desired LU factorization of A:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} \leftarrow -\mathbf{1} & 1 & 0 \\ m_{31} \leftarrow \mathbf{1} & m_{32} \leftarrow -\mathbf{1} & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 6 \\ 0 & 3 & 15 \\ 0 & 0 & 12 \end{bmatrix} = LU.$$

The operational count for triangularizing a matrix using Gaussian elimination is obtained in a way similar to the one we used for the naive Gaussian elimination. It can be shown that the number of multiplications and divisions needed for triangularizing an $n \times n$ matrix using Gaussian elimination is

$$\frac{n^3 - n}{3}$$

and the number of subtractions is

$$\frac{2n^3 - 3n^2 + n}{6}.$$

EXERCISE SET 4.4

1. Factor the following matrices into LU decomposition.

$$(a) \begin{bmatrix} 4 & -1 & 1 \\ 3 & 3 & 8 \\ 3 & 3 & 2 \end{bmatrix} \qquad (b) \begin{bmatrix} 3 & -1.5 & 2 \\ -1 & 0 & 2 \\ 4 & -3.5 & 5 \end{bmatrix}$$

$$(c) \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & -3 & 1 & 0 \\ 2 & -2 & 1 & 1 \end{bmatrix} \qquad (d) \begin{bmatrix} 2.75 & 4.23 & -2.88 & 5.45 \\ -4.45 & 6.00 & 0.00 & 1.19 \\ -1.00 & -5.33 & 1.11 & 0.00 \\ 6.34 & 7.00 & 0.00 & -4.54 \end{bmatrix}$$

2. Solve the following system using the LU decomposition.

(a) $$\begin{bmatrix} 3 & -2 & 1 & 0 \\ 1 & -4 & 2 & 2 \\ 1 & 0 & 3 & 0 \\ -3 & 7 & 9 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -3 \\ 6 \\ 5 \\ 5 \end{bmatrix}$$

(b) $$\begin{bmatrix} 1 & 1 & -2 \\ 4 & -2 & 1 \\ 3 & -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 8 \end{bmatrix}$$

3. The matrix

$$A = \begin{bmatrix} 2 & -1 & 2 \\ 2 & -3 & 3 \\ 6 & -1 & 8 \end{bmatrix}$$

has the following LU decomposition

$$L = \begin{bmatrix} 2 & 0 & 0 \\ 2 & -2 & 0 \\ 6 & 2 & 3 \end{bmatrix}, U = \begin{bmatrix} 1 & -\frac{1}{2} & 1 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}.$$

Use this decomposition of A to solve $A\mathbf{x} = \mathbf{b}$, where $\mathbf{b} = [-2, -5, 0]^T$.

4. The matrix A in the system of the linear equation

$$A\mathbf{x} = \mathbf{b}$$

has the LU decomposition

$$A = LU$$

with

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}, \quad U = L^T, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Determine $\mathbf{x}$.

5. The matrix A in the system of the linear equation

$$A\mathbf{x} = \mathbf{b}$$

has the LU decomposition

$$A = LU$$

with

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 \\ -1 & -1 & 0 & -1 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and $\mathbf{b} = [1, 1, 1, 1, 1]^T$. Determine $\mathbf{x}$.

6. Using Cholesky's method, calculate the decomposition $A = LL^T$, given

$$U = \begin{bmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{bmatrix}$$

7. Suppose Cholesky's method is applied to factor a matrix A. Determine A, if L has been found to be equal to

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 2 & 0 & 0 \\ -3 & -1 & 3 & 0 \\ 4 & 2 & 6 & 4 \end{bmatrix}$$

8. Consider the matrix

$$A = \begin{bmatrix} 2 & -3 & 2 & 5 \\ 1 & -1 & 1 & 2 \\ 3 & 2 & 2 & 1 \\ 1 & 1 & -3 & -1 \end{bmatrix}$$

Use Gaussian elimination to determine a unit lower triangular matrix M and an upper triangular matrix U such that $MA = U$.

9. **Computer assignment:**
Write a computer program in FORTRAN or C to solve a system of n linear equations and n unknowns using the LU decomposition.

Input data to the program should be

(a) The number of equations n.

(b) The augmented matrix.

Output should consist of:

(a) The augmented matrix.

(b) The L and U matrices.

(c) The vector solution.

Test your program to solve Exercise 1(b).

4.5 ITERATIVE METHODS

Because of round-off errors, direct methods become less efficient than iterative methods when they are applied to large systems, sometimes with as many as 100,000 variables. Examples of these large systems arise in the solution of partial differential equations. In these cases, an iterative method is preferable. In addition to round-off errors, the amount of storage space required for iterative solutions on a computer is far less than the one required for direct methods when the coefficient matrix of the system is sparse, that is, matrices which contain a high proportion of zeros. Thus, for sparse matrices, iterative methods are more attractive than direct methods.

An iterative scheme for linear systems consists of converting the system (4.1) to the form

$$\mathbf{x} = \mathbf{b}' - B\mathbf{x}.$$

After an initial guess, $\mathbf{x}^{(0)}$ is selected, the sequence of approximation solution vectors is generated by computing

$$\mathbf{x}^{(k)} = \mathbf{b}' - B\mathbf{x}^{(k-1)}$$

for each $k = 1, 2, 3, \ldots$.

4.5.1 Jacobi iterative method

We shall illustrate the method for a 3×3 linear system.

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = b_2$$
$$a_{31}x_1 + a_{32}x_2 + a_{33}x_3 = b_3$$

where we assume that the diagonal terms a_{11}, a_{22}, and a_{33} are all nonzero.

We begin our iterative scheme by solving each equation for one of the variables, choosing, when possible, to solve for the variable with the largest coefficient

$$\begin{aligned} x_1 &= & u_{12}x_2 &+ u_{13}x_3 + c_1 \\ x_2 &= u_{21}x_1 & &+ u_{23}x_3 + c_2 \\ x_3 &= u_{31}x_1 + u_{32}x_2 & & + c_3 \end{aligned}$$

where $u_{ij} = -\frac{a_{ij}}{a_{ii}}$, $c_i = \frac{b_i}{a_{ii}}$, $i = 1, 2, 3$.

Let $x_1^{(0)}$, $x_2^{(0)}$, $x_3^{(0)}$ be an initial approximation of the solution. The $(n+1)$st approximation is obtained from the approximation by writing

$$\begin{aligned} x_1^{(n+1)} &= & u_{12}x_2^{(n)} &+ u_{13}x_3^{(n)} + c_1 \\ x_2^{(n+1)} &= u_{21}x_1^{(n)} & &+ u_{23}x_3^{(n)} + c_2 \\ x_3^{(n+1)} &= u_{31}x_1^{(n)} + u_{32}x_2^{(n)} & & + c_3 \end{aligned}$$

for $n = 0, 1, 2, \ldots$.

In algorithmic form, the Jacobi iterative method may be presented as follows for an $n \times n$ linear system:

Consider (4.1) and solve for x_i in the ith equation to obtain, provided that $a_{ii} \neq 0$,

$$x_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} \left(-\frac{a_{ij}x_j}{a_{ii}} \right) + \frac{b_i}{a_{ii}}, \quad i = 1, 2, \ldots, n$$

and generate $x_i^{(k)}$ for $k \geq 1$ by

$$x_i^{(k)} = \frac{-\sum_{\substack{j=1 \\ j \neq i}}^{n} \left(a_{ij}x_j^{(k-1)} \right) + b_i}{a_{ii}}, \quad i = 1, 2, \ldots, n. \tag{4.11}$$

The iterative process is terminated when a convergence criterion is satisfied. One commonly used stopping criterion, known as the relative change criteria, is to iterate until

$$\frac{|\mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}|}{|\mathbf{x}^{(k)}|}, \qquad \mathbf{x}^{(k)} = \left(x_1^{(k)}, \ldots, x_n^{(k)} \right)^T$$

is less than a prescribed tolerance $\epsilon > 0$. Contrary to Newton's method for finding the roots of an equation, the convergence or divergence of the iterative process in the Jacobi method does not depend on the initial guess, but depends only on the character of the matrices themselves. However, a good first guess in case of convergence will make for a relatively small number of iterations. This is also true for the Gauss-Seidel method which will be presented in the next section.

EXAMPLE 4.10

Solve the following system using the Jacobi iterative method. Use $EPS = 10^{-3}$, $ITMAX = 30$, and $\mathbf{x}^{(0)} = \mathbf{0}$ as the starting vector.

$$\begin{aligned} 7x_1 - 2x_2 + x_3 \phantom{{}+ x_4} &= 17 \\ x_1 - 9x_2 + 3x_3 - x_4 &= 13 \\ 2x_1 \phantom{{}- 9x_2} + 10x_3 + x_4 &= 15 \\ x_1 - x_2 + x_3 + 6x_4 &= 10 \end{aligned}$$

These equations can be rearranged to give

$$\begin{aligned} x_1 &= (17 + 2x_2 - x_3)/7 \\ x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\ x_3 &= (15 - 2x_1 - x_4)/10 \\ x_4 &= (10 - x_1 + x_2 - x_3)/6 \end{aligned}$$

which provide the following Jacobi iterative process:

$$\begin{aligned} x_1^{(k+1)} &= (17 + 2x_2^{(k)} - x_3^{(k)})/7 \\ x_2^{(k+1)} &= (-13 + x_1^{(k)} + 3x_3^{(k)} - x_4^{(k)})/9 \\ x_3^{(k+1)} &= (15 - 2x_1^{(k)} - x_4^{(k)})/10 \\ x_4^{(k+1)} &= (10 - x_1^{(k)} + x_2^{(k)} - x_3^{(k)})/6 \end{aligned}$$

Substitute $\mathbf{x}^{(0)} = (0, 0, 0, 0)$ into the right-hand side of each of these equations to get

$$\begin{aligned} x_1^{(1)} &= (17 + 2(0) - 0)/7 = 2.428\,571\,429 \\ x_2^{(1)} &= (-13 + 0 + 3(0) - 0)/9 = -1.444\,444\,444 \\ x_3^{(1)} &= (15 - 2(0) - 0)/10 = 1.5 \\ x_4^{(1)} &= (10 - 0 + 0 - 0)/6 = 1.666\,666\,667 \end{aligned}$$

m-function 4.4

The following MATLAB function **jacobi.m** finds the solution of a linear system using the Jacobi iteration method. INPUTS are an *n x n* matrix *A;* an *n x 1* coefficient vector *b*; an initial vector x_0; a tolerance *tol*; the maximum number of iterations *itmax*.

```
function jacobi(A,b,x0,tol,itmax)
% Solve the system Ax=b using the Jacobi iteration method.
n=length(b);
x=zeros(n,1);
fprintf('\n');
disp('     The augmented matrix is =')
Augm=[A b]
Y=zeros(n,1);
Y=x0;
for k=1:itmax+1
  for i=1:n
    S=0;
    for j=1:n
      if (j~=i)
        S=S+A(i,j)*x0(j);
      end
    end
    if(A(i,i)==0)
      break
    end
    x(i)=(-S+b(i))/A(i,i);
  end
  err=abs(norm(x-x0));
  rerr=err/(norm(x)+eps);
  x0=x;
  Y=[Y x];
  if(rerr<tol)
    break
  end
end
% Print the results
if(A(i,i)==0)
 disp('       division by zero')
elseif (k==itmax+1)
 disp('       No convergence')
else
 fprintf('\n');
```

```
disp(' The solution vectors are:')
fprintf('\n');
disp('iter #     0       1       2       3       4       ...')
fprintf('\n');
for i=1:n
  fprintf('x%1.0f =    ',i)
  fprintf('%10.6f',Y(i,[1:k+1]))
  fprintf('\n');
end
fprintf('\n');
disp(['The method converges after ',num2str(k),' iterations to']);
x
end
```

and so $\mathbf{x}^{(1)} = (2.428\,571\,429, -1.444\,444\,444, 1.5, 1.666\,666\,667)^T$.

Similar procedure generates a sequence that converges to (see Table 4.6).

$$\mathbf{x}^{(9)} = (2.000127203, -1.000100162, 1.000118096, 1.000162172)^T$$

4.5.2 Gauss-Seidel iterative method

The algorithm for Gauss-Seidel is almost the same as for Jacobi, except that each x-value is improved using the most recent approximations to the values of the other variables. In this case, the $(n+1)$st approximation is obtained from the nth approximation for a 3×3 system by writing

$$\begin{aligned} x_1^{(n+1)} &= & u_{12}x_2^{(n)} &+ u_{13}x_3^{(n)} + c_1 \\ x_2^{(n+1)} &= u_{21}x_1^{(n+1)} & &+ u_{23}x_3^{(n)} + c_2 \\ x_3^{(n+1)} &= u_{31}x_1^{(n+1)} + u_{32}x_2^{(n+1)} & & + c_3 \end{aligned}$$

In algorithmic form, Gauss-Seidel may be presented as follows:
$x_i^{(k)}$ is generated for $k \geq 1$ by

$$x_i^{(k)} = \frac{-\sum_{j=1}^{i-1}(a_{ij}x_j^{(k)}) - \sum_{j=i+1}^{n}(a_{ij}x_j^{(k-1)}) + b_i}{a_{ii}} \tag{4.12}$$

for each $i = 1, 2, \ldots, n$.
The comments following the Jacobi algorithm regarding stopping criteria and starting vectors also apply to the Gauss-Seidel algorithm.

```
» A=[7 -2 1 0;1 -9 3 -1;2 0 10 1;1 -1 1 6];
» b=[17 13 15 10]';
» x0=[0 0 0 0]';
» jacobi(A,b,x0,10^(-3),30)

   The augmented matrix is =

Augm =

   7  -2   1   0  17
   1  -9   3  -1  13
   2   0  10   1  15
   1  -1   1   6  10

The solution vectors are:

iter #      0          1          2          3          4          . .

x1 =    0.000000   2.428571   1.801587   2.061829   1.977515   2.008259   1.997187   2.001020   1.999639   2.000127
x2 =    0.000000  -1.444444   0.859788  -1.047413  -0.981367  -1.006166  -0.997771  -1.000802  -0.999721  -1.000100
x3 =    0.000000   1.500000   0.847619   1.062566   0.979451   1.007360   0.997320   1.000926   0.999667   1.000118
x4 =    0.000000   1.666667   0.771164   1.081834   0.971365   1.010278   0.996369   1.001287   0.999542   1.000162

The method converges after 9 iterations to

x =
            2.0001
           -1.0001
            1.0001
            1.0002
```

Table 4.6 The Jacobi iterative method for Example 4.10.

Because the new values can be immediately stored in the location that held the old values, the storage requirements for $\mathbf{x}$ with the Gauss-Seidel method is half what it would be with the Jacobi method and the rate of convergence is more rapid.

EXAMPLE 4.11

Solve the following system using the Gauss-Seidel iterative method. Use $EPS = 10^{-3}$, $ITMAX = 30$ and $\mathbf{x}^{(0)} = \mathbf{0}$ as the starting vector.

$$\begin{aligned} 7x_1 - 2x_2 + x_3 \phantom{{}+ x_4} &= 17 \\ x_1 - 9x_2 + 3x_3 - x_4 &= 13 \\ 2x_1 \phantom{{}- 2x_2} + 10x_3 + x_4 &= 15 \\ x_1 - x_2 + x_3 + 6x_4 &= 10 \end{aligned}$$

From Example 4.10, we have

$$x_1 = (17 + 2x_2 - x_3)/7$$

m-function 4.5

The following MATLAB function **seidel.m** finds the solution of a linear system using Gauss-Seidel iteration method. INPUTS are an *n x n* matrix *A;* an *n x 1* coefficient vector *b*; an initial vector x_0; a tolerance *tol*; the maximum number of iterations *itmax.*

```
function seidel(A,b,x0,tol,itmax)
% Solve the system Ax=b using Gauss-Seidel iteration method.
n=length(b);
x=zeros(n,1);
fprintf('\n');
disp('     The augmented matrix is =')
Augm=[A b]
Y=zeros(n,1);
Y=x0;
for k=1:itmax+1
  for i=1:n
    S=0;
    for j=1:i-1
      S=S+A(i,j)*x(j);
    end
    for j=i+1:n
      S=S+A(i,j)*x0(j);
    end
    if(A(i,i)==0)
      break
    end
    x(i)=(-S+b(i))/A(i,i);
  end
  err=abs(norm(x-x0));
  rerr=err/(norm(x)+eps);
  x0=x;
  Y=[Y x];
  if(rerr<tol)
    break
  end
end
% Print the results
if(A(i,i)==0)
 disp('      division by zero')
elseif (k==itmax+1)
 disp('      No convergence')
else
```

```
fprintf('\n');
disp(' The solution vectors are:')
fprintf('\n');
disp('iter #      0       1       2       3       4       ...')
fprintf('\n');
for i=1:n
  fprintf('x%1.0f =    ',i)
  fprintf('%10.6f',Y(i,[1:k+1]))
  fprintf('\n');
end
fprintf('\n');
disp(['The method converges after ',num2str(k),' iterations to']);
x
end
```

$$
\begin{aligned}
x_2 &= (-13 + x_1 + 3x_3 - x_4)/9 \\
x_3 &= (15 - 2x_1 - x_4)/10 \\
x_4 &= (10 - x_1 + x_2 - x_3)/6
\end{aligned}
$$

which provide the following Gauss-Seidel iterative process:

$$
\begin{aligned}
x_1^{(k+1)} &= (17 + 2x_2^{(k)} - x_3^{(k)})/7 \\
x_2^{(k+1)} &= (-13 + x_1^{(k+1)} + 3x_3^{(k)} - x_4^{(k)})/9 \\
x_3^{(k+1)} &= (15 - 2x_1^{(k+1)} - x_4^{(k)})/10 \\
x_4^{(k+1)} &= (10 - x_1^{(k+1)} + x_2^{(k+1)} - x_3^{(k+1)})/6
\end{aligned}
$$

Substitute $\mathbf{x}^{(0)} = (0, 0, 0, 0)$ into the right-hand side of each of these equations to get

$$
\begin{aligned}
x_1^{(1)} &= (17 + 2(0) - 0)/7 = 2.428\,571\,429 \\
x_2^{(1)} &= (-13 + 2.428\,571\,429 + 3(0) - 0)/9 = -1.1746031746 \\
x_3^{(1)} &= (15 - 2(2.428\,571\,429) - 0)/10 = 1.0142857143 \\
x_4^{(1)} &= (10 - 2.428\,571\,429 - 1.1746031746 - 1.0142857143)/6 \\
&= 0.8970899472
\end{aligned}
$$

and so

$$
\mathbf{x}^{(1)} = (2.428571429, -1.1746031746, 1.0142857143, 0.8970899472)^T.
$$

```
» A=[7 -2 1 0;1 -9 3 -1;2 0 10 1;1 -1 1 6];
» b=[17 13 15 10]';
» x0=[0 0 0 0]';
» seidel(A,b,x0,10^(-3),30)

    The augmented matrix is

Augm =

   7  -2   1   0  17
   1  -9   3  -1  13
   2   0  10   1  15
   1  -1   1   6  10

 The solution vectors are:

iter #    0          1          2          3          4          ...

x1 =    0.000000   2.428571   1.948073   2.000025   2.001849   2.000025
x2 =    0.000000  -1.174603  -0.989573  -0.993877  -1.000154  -1.000130
x3 =    0.000000   1.014286   1.020676   0.999300   0.999517   1.000020
x4 =    0.000000   0.897090   1.006946   1.001133   0.999747   0.999971

The method converges after 5 iterations to

 x =
            2.0000
           -1.0001
            1.0000
            1.0000
```

Table 4.7 The Gauss-Seidel iterative method for Example 4.11.

Similar procedure generates a sequence that converges to (see Table 4.7).

$$\mathbf{x}^{(5)} = (2.000025, -01.000130, 1.000020, 0.999971)^T$$

4.5.3 Convergence

The matrix formulation of the Jacobi and Gauss-Seidel iterative methods can be obtained by splitting the matrix A into the sum

$$A = D + L + U$$

where D is the diagonal of A, L the lower triangular part of A, and U the upper triangular part of A. That is,

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}, \; L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}, \; U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ 0 & \cdots & 0 & 0 \end{bmatrix}.$$

Thus, the system (4.1) can be written as

$$(D + L + U)\,\mathbf{x} = \mathbf{b}.$$

The Jacobi method in matrix form is

$$D\,\mathbf{x}^{(k)} = -(L + U)\,\mathbf{x}^{(k-1)} + \mathbf{b}$$

and the Gauss-Seidel method in matrix form is

$$(D + L)\,\mathbf{x}^{(k)} = -U\,\mathbf{x}^{(k-1)} + \mathbf{b}.$$

Before stating the theorem on the convergence of the Jacobi and Gauss-Seidel methods, we make the following definition:

DEFINITION 4.3 *An $n \times n$ matrix A is* **strictly diagonally dominant** *if*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \textit{for } i = 1, 2, \ldots, n$$

We now give a sufficient condition for Jacobi and Gauss-Seidel to convergence.

THEOREM 4.1 (Jacobi and Gauss-Seidel convergence theorem)
If A is strictly diagonally dominant, then the Jacobi and Gauss-Seidel methods converge for any choice of the starting vector $\mathbf{x}^{(0)}$.

Proof: The proof can be found in advanced texts on numerical analysis.

EXAMPLE 4.12
Consider the system of equation

$$\begin{bmatrix} 3 & 1 & 1 \\ -2 & 4 & 0 \\ -1 & 2 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 2 \end{bmatrix}$$

The coefficient matrix of the system is strictly diagonally dominant since

$$
\begin{aligned}
|a_{11}| &= |3| = 3 > |1| + |1| = 2 \\
|a_{22}| &= |4| = 4 > |-2| + |0| = 2 \\
|a_{33}| &= |-6| = 6 > |-1| + |2| = 3.
\end{aligned}
$$

Hence, if the Jacobi or Gauss-Seidel method is used to solve the system of equations, then it will converge for any choice of the starting vector $\mathbf{x}^{(0)}$.

EXERCISE SET 4.5

1. Solve the following systems using the Gauss-Seidel iterative method

(a)
$$
\begin{aligned}
x - y + 2z - w &= -1 \\
2x + y - 2z - 2w &= -2 \\
-x + 2y - 4z + w &= 1 \\
3x - 3w &= -3
\end{aligned}
$$

(b)
$$
\begin{aligned}
5x - y + 3z &= 3 \\
4x + 7y - 2z &= 2 \\
6x - 3y + 9z &= 9
\end{aligned}
$$

2. Consider the linear system

$$
\begin{aligned}
x_1 + 4x_2 &= -15 \\
5x_1 + x_2 &= 1
\end{aligned}
$$

Apply the Jacobi method to this arrangement, beginning with a vector close to the solution $\mathbf{x} = [1.01, -4.01]$ and observe divergence. Now interchange the equations to solve the system again and observe convergence.

3. Solve the following system using the Jacobi method

$$\begin{bmatrix} 17 & 1 & 4 & 3 & -1 & 2 & 3 & -7 \\ 2 & 10 & -1 & 7 & -2 & 1 & 1 & -4 \\ -1 & 1 & -8 & 2 & -5 & 2 & -1 & 1 \\ 2 & 4 & 1 & -9 & 1 & 3 & 4 & -1 \\ 1 & 3 & 1 & 7 & 15 & 1 & -2 & 4 \\ -2 & 1 & 7 & -1 & 2 & 12 & -1 & 8 \\ 3 & 4 & 5 & 1 & 2 & 8 & -8 & 2 \\ 5 & 1 & 1 & 1 & -1 & 1 & -7 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} 71 \\ 43 \\ -11 \\ -33 \\ -65 \\ 52 \\ -73 \\ 20 \end{bmatrix}$$

4. The solution of the system

$$\begin{aligned} x_1 + 3x_2 &= 3 \\ 4x_1 + \ \ x_2 &= 1 \end{aligned}$$

is $x_1 = 0$ and $x_2 = 1$. Apply the Jacobi and Gauss-Seidel methods to this rearrangement, starting with a vector close to the solution. Which method diverges most rapidly?

5. Show that the matrix

$$A = \begin{bmatrix} 16 & -1 & 4 & 0 \\ 2 & 9 & 1 & 1 \\ 3 & 0 & 11 & 2 \\ 1 & 1 & 0 & 13 \end{bmatrix}$$

is positive definite, then solve the linear system

$$A\mathbf{x} = \mathbf{b} \quad \text{with} \quad \mathbf{b} = [1, -1, 2, 0]^T$$

using the Gauss-Seidel iterative method.

6. Given the system

$$\begin{aligned} x \qquad\quad + \ z &= 2 \\ x - \ y \qquad\quad &= 0 \\ x + 2y - 3z &= 0 \end{aligned}$$

use the MATLAB functions seidel and jacobi to check that the Jacobi iterative method converges whereas the Gauss-Seidel iterative method diverges. Use $TOL = 10^{-3}$ and start with $\mathbf{x}^{(0)} = \mathbf{0}$.

7. **Computer assignment:**
Write a computer program in FORTRAN or C to solve a system of n linear equations and n unknowns using:

(a) The Jacobi iterative method.
(b) The Gauss-Seidel iterative method.

Input data to the program should be

(a) The number of equations n.
(b) The augmented matrix.
(c) The starting vector.

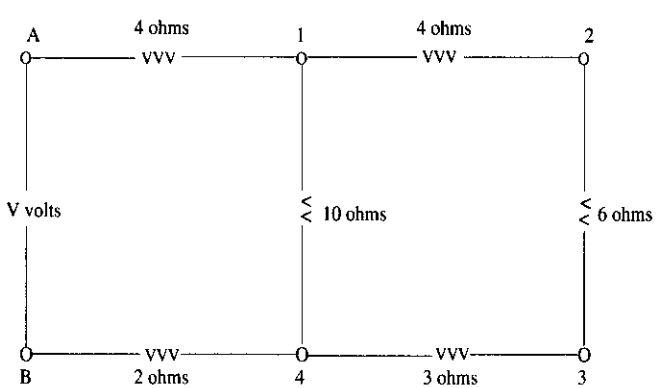
Output should consist of:

(a) The augmented matrix.
(b) The vector solution.

Test your program to solve Exercise 1(a).

APPLIED PROBLEMS FOR CHAPTER 4

1. Consider the following electrical network:

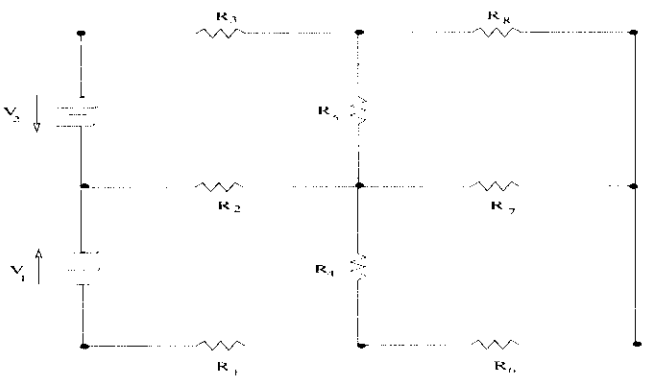


By applying both Ohm's law and Kirchoff's Current law, we get the following system of linear equations

$$\begin{aligned} 3v_1 - 5v_2 + 2v_3 &= 0 \\ -2v_1 + 5v_2 + 2v_4 &= -5V \\ v_2 - 3v_3 + 2v_4 &= 0 \\ 3v_1 + 10v_3 - 28v_4 &= 0 \end{aligned}$$

where $v_1, \ldots, v_4$ denote the potentials and V the potential applied between A and B.
Solve the system for $V = 40, 80$.

2. A resistor network with two voltage sources is shown below. By



applying both Ohm's law and Kirchoff's Current law, we get the following system of linear equations

$$\begin{bmatrix} R_1 + R_2 + R_4 & -R_2 & 0 & -R_4 \\ -R_2 & R_2 + R_3 + R_5 & -R_5 & 0 \\ 0 & -R_5 & R_5 + R_7 + R_8 & -R_7 \\ -R_4 & 0 & -R_7 & R_4 + R_6 + R_7 \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} -V_1 \\ V_2 \\ 0 \\ 0 \end{bmatrix}$$

Solve the system of equation if $R_1 = R_2 = R_3 = 100$, $R_4 = R_5 = 150$, $R_6 = R_7 = R_8 = 200$, $V_1 = 10$, and $V_2 = 12$.
Note: The coefficient matrix is diagonally dominant.

3. Least square polynomial consists of finding a polynomial that best represents a set of experimental data. A polynomial of degree three $y = a_0 + a_1 x + a_2 x^2 + a_3^3$ that best fits n data points, is obtained from the solution of the system of equations (see Chapter 7):

$$\begin{aligned} a_0 n \quad\quad + a_1 \sum x_i + a_2 \sum x_i^2 + a_3 \sum x_i^3 &= \sum y_i \\ a_0 \sum x_i + a_1 \sum x_i^2 + a_2 \sum x_i^3 + a_3 \sum x_i^4 &= \sum y_i x_i \\ a_0 \sum x_i^2 + a_1 \sum x_i^3 + a_2 \sum x_i^4 + a_3 \sum x_i^5 &= \sum y_i x_i^2 \\ a_0 \sum x_i^3 + a_1 \sum x_i^4 + a_2 \sum x_i^5 + a_3 \sum x_i^6 &= \sum y_i x_i^3 \end{aligned}$$

Find the least square polynomial of degree three that best fits to the following data

x	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5
y	4.3	6.2	10.1	13.5	19.8	22.6	24.7	29.2

4. Consider the loading of statically determinate pin-jointed truss shown in Figure 4.1. The truss has seven members and five nodes, and is under the action of the forces R_1, R_2, and R_3 parallel to the y-axis.

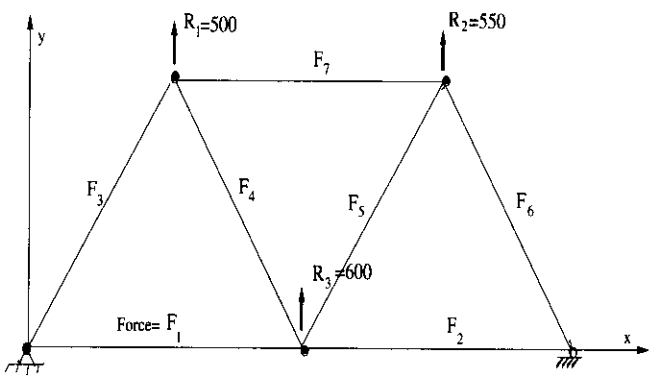
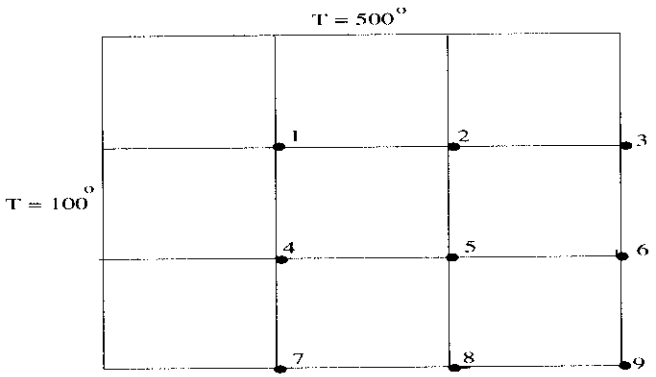


FIGURE 4.1
Determinate pin-jointed truss.

Using the result that at each pin, the sum of all forces Fi acting horizontally of vertically is equal to zero, find the member forces (Fi) obtained from the following system of equations.

$$\begin{bmatrix} 1.0 & -1.0 & 0 & 0.5 & -0.5 & 0 & 0 \\ 0 & 0 & 0 & -8.66 & -8.66 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & -0.5 & 0 & 0 & -1.0 \\ 0 & 0 & 8.66 & 8.66 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.5 & -0.5 & 1.0 \\ 0 & 0 & 0 & 0 & 8.66 & 8.66 & 0 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \\ F_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 500 \\ 0 \\ 0 \\ 550 \\ 0 \\ 600 \end{bmatrix}$$

5. Consider the square shown below. The left face is maintained at 100°C and the top face at 500°C, while the other faces are exposed to an environment at 100°C. By applying a numerical method known as the finite difference, the temperature at the various nodes is given from the solution of the system of equations:



$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & -4.67 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 2 & -4.67 & 0 & 0 & 1 \\ 0 & 0 & 0 & 2 & 0 & 0 & -4.67 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 1 & -4.67 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -2.67 \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \end{bmatrix} = \begin{bmatrix} -600 \\ -500 \\ -567 \\ -100 \\ 0 \\ -67 \\ -167 \\ -67 \\ -67 \end{bmatrix}$$

Find the temperature T_i, $i = 1, \ldots, 9$.

Note: The coefficient matrix is sparse (contains a large number of zero elements) and diagonally dominant. For this reason, iterative methods of solution may be very efficient.

Chapter 5

Interpolation

Consider the following problem:

x	x_0	x_1	$\cdots$	x_n
y	y_0	y_1	$\cdots$	y_n

Table 5.1 Table of numerical values.

Suppose that we are given a Table of $(n+1)$ points where the $x's$ are distinct and satisfy

$$x_0 < x_1 < x_2 < \ldots < x_n.$$

Our objective is to find a polynomial curve that passes through the given points (x_i, y_i), $i = 0, 1, \ldots, n$. Hence, we need to find a polynomial $p(x)$ such that

$$p(x_i) = y_i \quad \text{for} \quad i = 0, 1, \ldots, n. \tag{5.1}$$

This is a topic called **interpolation**. The polynomial $p(x)$ is said to interpolate Table 5.1 and the points x_i are called the **nodes**.

Two methods for doing this will be developed in this chapter. The first is **Newton's method**, based on using the *divided differences* and the second is the **Lagrange method**, based on using the so-called *cardinal functions*.

5.1 POLYNOMIAL INTERPOLATION THEORY

Given the data in Table 5.1 where the $x_i's$ are assumed to be distinct, we want to study the problem of finding a polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{5.2}$$

that interpolates the given data.

Applying condition (5.1) to the given tabulated points x_i gives the system

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1 \\ \vdots \qquad\qquad\qquad\qquad &= \vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n \end{aligned} \tag{5.3}$$

This is a system of $(n+1)$ linear equations in $(n+1)$ unknowns $a_0, a_1, \ldots, a_n$. In matrix form, the system is

$$\mathbf{X}a = \mathbf{Y}$$

where

$$\mathbf{X} = \left[x_i^j\right] \qquad i, j = 0, 1, \ldots, n$$

$$a = [a_0, \ldots, a_n]^T, \qquad \mathbf{Y} = [y_0, \ldots, y_n]^T \tag{5.4}$$

The matrix $\mathbf{X}$ is known as the **Vandermonde** matrix.
Thus, solving the system (5.3) is equivalent to solving the polynomial interpolation problem.

THEOREM 5.1
Given $n+1$ distinct points $x_0, x_1, \ldots, x_n$ and $n+1$ arbitrary real values $y_0, y_1, \ldots, y_n$, there is a unique polynomial p of degree $\leq n$ that interpolates Table 5.1.

Proof: It can be easily shown that the determinant of the matrix $\mathbf{X}$ in (5.4) is (see Exercise 2)

$$\det(\mathbf{X}) = \prod_{0 \leq j < i \leq n} (x_i - x_j) \tag{5.5}$$

and is nonzero since the points x_i are assumed to be distinct. Thus, there is a unique solution for the a_i's, that is, there is a unique interpolating polynomial of degree $\leq n$.

EXAMPLE 5.1

Find the polynomial that interpolates the table

x	4.5	6.1
y	7.1	2.3

The polynomial sought has the form

$$p(x) = a_0 + a_1 x$$

By applying condition (5.1) to the given table, we obtain the system

$$\begin{aligned} a_0 + 4.5a_1 &= 7.1 \\ a_0 + 6.1a_1 &= 2.3. \end{aligned}$$

Solve to get

$$a_0 = 20.6, \quad a_1 = -3.0.$$

Therefore,

$$p(x) = 20.6 - 3.0x$$

EXERCISE SET 5.1

1. Find a polynomial of order 2 that interpolates the table

x	0.2	0.4	0.6
y	-.95	-.82	-.65

2. Show the determinant of the Vandermonde matrix

$$V = (x_0, x_1, \ldots, x_n) = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ x_0 & x_1 & x_2 & \ldots & x_n \\ x_0^2 & x_1^2 & x_2^2 & \ldots & x_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ x_0^n & x_1^n & x_2^n & \ldots & x_n^n \end{bmatrix} = \prod_{0 \le j < i \le n} (x_i - x_j)$$

3. Determine a polynomial of degree ≤ 3 that interpolates the data

x	1.2	2.1	3.0	3.6
y	0.7	8.1	27.7	45.1

5.2 NEWTON'S DIVIDED DIFFERENCE INTERPOLATING POLYNOMIAL

In this section we want to develop an interpolating formula which will be more efficient and convenient to use than the one shown in the previous section. This will avoid the problem of finding the solution to a system of simultaneous equations.

Let us first derive the first-degree Newton interpolating polynomial ($n = 1$) which takes in this case, the form

$$p_1(x) = a_0 + a_1(x - x_0)$$

Since a straight line can be passed through two points, then

$$p_1(x) = y_0 + \left(\frac{y_1 - y_0}{x_1 - x_0}\right)(x - x_0) \tag{5.6}$$

At this stage, we introduce the following notation

$$f[x_0, x_1, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0} \tag{5.7}$$

where

$$y_i = f(x_i) = f[x_i], \qquad i = 0, \ldots, n$$

This coefficient is called the nth **order divided difference** of f. In view of (5.7), $p_1(x)$ can be written in the form

$$p_1(x) = f[x_0] + f[x_0, x_1](x - x_0)$$

or

$$p_1(x) = p_0(x) + f[x_0, x_1](x - x_0). \tag{5.8}$$

Note that $p_1(x)$ is obtained by adding to $p_0(x)$ the correction term

$$f[x_0, x_1](x - x_0).$$

We now proceed to the second-degree interpolating polynomial. From Eqn. (5.8) we would expect $p_2(x)$ to be in the form

$$p_2(x) = p_1(x) + a_2(x - x_0)(x - x_1). \tag{5.9}$$

Note that, for each of the points x_0 and x_1, condition (5.1) is satisfied, since $p_1(x)$ itself satisfies (5.1) and $(x - x_0)(x - x_1)$ takes the value zero.

Evaluating $p_2(x)$ at $x = x_2$ and using the fact that $p_2(x_2) = f[x_2]$, we find that (see Exercise 8)

$$a_2 = \left[\frac{f(x_2) - f(x_1)}{x_2 - x_1} - \frac{f(x_1) - f(x_0)}{x_1 - x_0}\right] / (x_2 - x_0)$$

Using Eqn. (5.7), a_2 can be expressed as

$$\begin{aligned} a_2 &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ &= f[x_0, x_1, x_2]. \end{aligned}$$

Thus, in view of (5.9), we have

$$p_2(x) = p_1(x) + f[x_0, x_1, x_2](x - x_0)(x - x_1). \tag{5.10}$$

Repeating this entire process again, p_3, p_4, and higher interpolating polynomials can be consecutively obtained in the same way. In general, we would obtain for $p_n(x)$, the interpolating formula

$$p_n(x) = p_{n-1}(x) + f\left[x_0, \ldots, x_n\right](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \tag{5.11}$$

We now turn to the problem of calculating the divided differences. We start with Table 5.2 which gives the values of a function f where the

x	x_0	x_1	$\dots$	x_n
$f(x)$	$f(x_0)$	$f(x_1)$	$\dots$	$f(x_n)$

Table 5.2 Values of a function f.

points $x_0, \dots, x_n$ are assumed to be distinct.

Using Eqn. (5.7), the first four divided differences are

$$\begin{aligned} f[x_0] &= f(x_0) \\ f[x_0, x_1] &= \frac{f[x_1] - f[x_0]}{x_1 - x_0} \\ f[x_0, x_1, x_2] &= \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \\ f[x_0, x_1, x_2, x_3] &= \frac{f[x_1, x_2, x_3] - f[x_0, x_1, x_2]}{x_3 - x_0}. \end{aligned}$$

One should note that $f[x_0, x_1, \dots, x_n]$ is invariant under all permutations of the arguments $x_0, x_1, \dots, x_n$. More general properties of divided differences may be found in [5].

This procedure is best illustrated by arranging the divided differences in the format shown in Table 5.3.

The coefficient in the top diagonal of the table are the ones needed to write the Newton interpolating polynomial (5.11), that is

$$\begin{aligned} p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots \\ \cdots + f[x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \cdots (x - x_{n-1}). \end{aligned}$$

which can be expressed in the more compact form

$$p_n(x) = \sum_{i=0}^{n} f[x_0, x_1, \dots, x_i] \prod_{j=1}^{i} (x - x_{j-1}). \tag{5.12}$$

Formula (5.12) is known as the **Newton forward divided-difference** formula. If the nodes are recorded as $x_n, x_{n-1}, \dots, x_0$, a similar formula known as the **Newton backward divided-difference** formula is obtained

$$p_n(x) = \sum_{i=0}^{n} f[x_{n-i}, x_{n-i+1}, \dots, x_n] \prod_{j=1}^{i} (x - x_{n-j+1}) \tag{5.13}$$

x_i	$f[x_i]$	$f[x_i, x_{i+1}]$	$f[x_i, x_{i+1}, x_{i+2}]$ ...
x_0	$f[x_0]$		
		$f[x_0, x_1]$	
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$
		$f[x_1, x_2]$	
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$
		$f[x_2, x_3]$	
x_3	$f[x_3]$		$f[x_2, x_3, x_4]$
		$f[x_3, x_4]$	
x_4	$f[x_4]$		$f[x_3, x_4, x_5]$
		$f[x_4, x_5]$	
x_5	$f[x_5]$		
$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 5.3 Format for constructing divided differences of $f(x)$.

EXAMPLE 5.2

Form a divided-difference table for the following data and obtain Newton's interpolating polynomial.

x	2	4	6	8
y	4	8	14	16

From the top diagonal of Table 5.4 Newton's interpolating polynomial is

$$p_3(x) = 4 + 2(x-2) + \frac{1}{4}(x-2)(x-4) + \frac{1}{8}(x-2)(x-4)(x-6).$$

When the nodes $x_0, x_1, \ldots, x_n$ are evenly spaced, that is,

$$x_k = x_0 + kh, \quad k = 0, 1, \ldots, n$$

x_i	$f[x_i]$	f[,]	f[,,]	f[,,,]
2	4			
		$\frac{8-4}{4-2} = 2$		
4	8		$\frac{3-2}{6-2} = \frac{1}{4}$	
		$\frac{14-8}{6-4} = 3$		$\frac{\frac{1}{4}+\frac{1}{2}}{8-2} = \frac{1}{8}$
6	14		$\frac{1-3}{8-4} = -\frac{1}{2}$	
		$\frac{16-14}{8-6} = 1$		
8	16			

Table 5.4 Divided differences for Example 5.2.

with

$$h = \frac{x_n - x_0}{n}$$

Eqns. (5.12) and (5.13) can be expressed in a simplified form as follows:

Let us assume that the nodes are arranged in ascending order. We define the **forward difference** Δf_i by

$$\Delta f_i = f_{i+1} - f_i, \quad \text{with } f(x_i) = f_i, \quad i = 0, 1, \ldots$$

Higher powers are defined recursively by

$$\Delta^n f_i = \Delta\left(\Delta^{n-1} f_i\right), \qquad n = 2, 3, \ldots$$

Using the binomial-coefficient notation

$$\binom{s}{k} = \frac{s(s-1)\cdots(s-k+1)}{k!}$$

Eqn. (5.12) can be expressed as

$$p_n(x) = \sum_{k=0}^{n} \binom{s}{k} \Delta^k f_0, \qquad s = \frac{x - x_0}{h}. \tag{5.14}$$

Eqn. (5.14) is called the **Newton forward difference**. Similarly Eqn. (5.13) can be expressed as

$$p_n(x) = \sum_{k=0}^{n} (-1)^k \binom{-s}{k} \nabla^k f_n, \tag{5.15}$$

with $\nabla f_i = f_i - f_{i-1}, i \geq 1$. Eqn. (5.15) is called the **Newton backward difference**. For more details regarding the derivation of these formulas see [14].

EXAMPLE 5.3

Form a divided-difference table for the following data and obtain Newton's interpolating polynomial.

x	1	2	3	5	7
y	3	5	9	11	15

```
» x=[1 2 3 5 7]';
» y=[3 5 9 11 15]';
» newtondd(x,y)

        Newton divided difference
```

x	y	f[,]	f[,,]	f[,,,]	...
1.00	3.00				
		2.00			
2.00	5.00		1.00		
		4.00		-0.50	
3.00	9.00		-1.00		0.125
		1.00		0.25	
5.00	11.00		0.25		
		2.00			
7.00	15.00				

Table 5.5 Newton's divided-difference for Example 5.3.

The MATLAB function newtondd produces the coefficients for the polynomial which appears in Table 5.5. The data and polynomial are shown in Figure 5.1. The equation of Newton's interpolating polynomial is

$$p_4(x) = \underline{3} + \underline{2}(x-1) + (x-1)(x-2) - \frac{\underline{1}}{\underline{2}}(x-1)(x-2)(x-3)$$
$$+ \frac{\underline{1}}{\underline{8}}(x-1)(x-2)(x-3)(x-5)$$

m-function 5.1

The following MATLAB function **newtondd.m** forms a divided-difference table for the Newton interpolating polynomial. INPUTS are the vectors *x* and *y* .

```
function newtondd(x,y)
% Newton divided difference
disp('            Newton divided difference')
disp('_____________________________________________________')
disp('   x          y          f[,]          f[,,]          f[,,,]          ... ')
disp('_____________________________________________________')
n=length(x);
for k=1:n-1
 % Compute first divided difference
   d(k,1)=(y(k+1)-y(k))/(x(k+1)-x(k));
end
for i=2:n-1
   for k=1:n-i
  % Compute ith divided difference
      d(k,i)=(d(k+1,i-1)-d(k,i-1))/(x(k+i)-x(k));
   end
end
% print results
if (rem(n,2)==0)
   p=n/2;
   m=n/2;
else
   p=fix(n/2);
   m=fix(n/2)+1;
end
for i=1:p
  fprintf('%8.2f %8.2f',x(i),y(i));
  for k=1:i-1
     fprintf('            %8.5f',d(i-k,2*k));
  end
  fprintf('\n     ')
  for k=1:i
     fprintf('            %8.5f',d(i-k+1,2*k-1));
  end
  fprintf('\n')
end
j=p;
for i=m:-1:1
```

```
    j=j+1;
    fprintf('%8.2f %8.2f',x(j),y(j));
    for k=1:i-1
      fprintf('          %8.5f',d(j-k,2*k));
    end
    fprintf('\n     ')
    for k=1:i-1
      fprintf('          %8.5f',d(j-k+1,2*k-1));
    end
    fprintf('\n')
end
```

EXAMPLE 5.4

The following table lists values of the sine function at various points. The divided-difference table corresponding to these data is shown in Table 5.6.

x	0.0	0.1	0.3	0.7	0.8	0.9
$\sin x$	0.0	0.099833	0.29552	0.644218	0.717356	0.783327

x	1.3	1.9	2.2	2.7	3.1
$\sin x$	0.963558	0.94630	0.808496	0.427380	0.041581

EXERCISE SET 5.2

1. Determine the polynomial of degree ≤ 3 that interpolates the function $f(x) = 2x^2 - x + 2$ at $x = 0, 1, 2, 3$. What is the degree of the polynomial?

2. Show that Newton's interpolating polynomial of degree, one passing through the points (x_0, y_0) and (x_1, y_1), may be written in the form

$$p_1(x) = \frac{1}{x_1 - x_0} \begin{vmatrix} y_0 & x_0 - x \\ y_1 & x_1 - x \end{vmatrix}$$

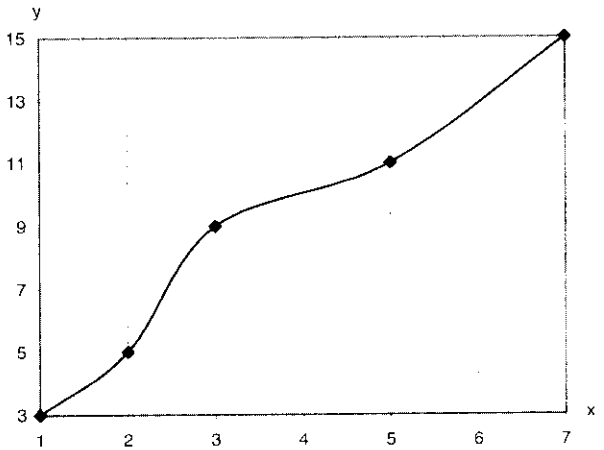


FIGURE 5.1
Interpolating polynomial for Example 5.3.

3. Determine the polynomial of degree ≤ 5, using Newton's divided-differences, that interpolates the table

x	1.0	2.0	3.0	4.0	5.0	6.0
$f(x)$	14.5	19.5	30.5	53.5	94.5	159.5

Use the resulting polynomial to estimate the value of $f(4.5)$. Compare with the exact value, $f(4.5) = 71.375$.

4. Construct Newton's interpolating polynomial for the data given by the table

x	-3	-2	-1	0	1
y	-41	-17	-5	1	3

5. Fill in the divided difference table:

```
» x=[0 0.1 0.3 0.7 0.8 0.9 1.3 1.9 2.2 2.7 3.1]';
» y=sin(x);
» newtondd(x,y)
```

Newton divided difference

x	y	f[.]	f[..]	f[...]	...					
0	0									
		0.99833								
0.1	0.099833		-0.0663167							
		0.978435		-0.1592857						
0.3	0.295520		-0.1778167		0.01533333					
		0.871745		-0.147019		0.00748016				
0.7	0.644218		-0.28073		0.02206548		-0.0009829			
		0.73138		-0.1293667		0.00620238		0.00006		
0.8	0.717356		-0.35835		0.02950833		-0.0008633		-0.00013	
		0.65971		-0.0998583		0.00464836		-0.00022		0.00008
0.9	0.783327		-0.418265		0.03694571		-0.0013179		0.00008	
		0.4505775		-0.0555235		0.00214441		0.00000		-0.00003
1.3	0.963558		-0.4793408		0.04016233		-0.0013202		0.00001	
		-0.0287633		0.00070377		-0.0004961		0.00001		
1.9	0.946300		-0.4784259		0.03921983		-0.0012855			
		-0.4593467		0.07129947		-0.0034526				
2.2	0.808496		-0.3786067		0.03162401					
		-0.762232		0.12822269						
2.7	0.427380		-0.2247394							
		-0.9644975								
3.1	0.041581									

Table 5.6 Newton's divided-differences for Example 5.4.

x_i	$f[.]$	$f[.,.]$	$f[.,.,]$	$f[.,.,.,.]$
1.1	2.45			
		0.609		
2.2	3.12		—	
		—		—
3.3	3.98		—	
		—		
4.4	5.22			

6. Given $f(x) = 2x^3 + 4x^2 + 3x + 5$. Find Newton's interpolating polynomial that interpolates the values of f at $x = -1, 0, 1, 2, 3$.

7. To investigate the relationship between yield of potatoes, y, and level of fertilizer application, x, an experimenter divided a field into 5 plots of equal size and applied differing amounts of fertilizer to each. The data recorded for each plot are given by the table (in pounds).

x	1	2	3	4	5
y	22	23	25	30	28

(a) Find an interpolating polynomial for this table.

(b) According to the interpolating polynomial, approximately how many pounds would you expect from a plot to which 2.5 pounds of fertilizer had been applied?

8. Show that the coefficient a_2 in Eqn. (5.9) is equal to

$$a_2 = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

9. **Computer assignment:**
Write a computer program in FORTRAN or C that performs Newton interpolation. Input data to the program should include a set of values of x, the number of such values, the corresponding values y, and the value α of x at which Newton's interpolating polynomial, $p(x)$, is to be evaluated. The output should be the divided-difference table and the value of $p(\alpha)$.

Test your program using the data in Exercise 4. Find $p(-1.6)$.

5.3 THE ERROR OF THE INTERPOLATING POLYNOMIAL

Let us now derive an expression for the error term $f(x) - p_n(x)$. In general, an error exists at the nontabular points, known as the truncation error. To determine the form of this error term, we consider a function $f(x)$ which has $n+1$ continuous derivatives in $[a, b]$. Let the error term be represented by

$$E_n(x) = f(x) - p_n(x). \tag{5.16}$$

We will now prove that for any t in $[a, b]$ containing the distinct nodes $x_0, x_1, \ldots, x_n$, there is a ζ in (a, b) for which

$$E_n(t) = \frac{1}{(n+1)!} f^{(n+1)}(\zeta) \prod_{i=0}^{n} (t - x_i). \tag{5.17}$$

Let t be fixed and distinct from the nodal points x_i. To simplify the notation, we define

$$\Psi(x) = \prod_{i=0}^{n}(x - x_i) \tag{5.18}$$

so that

$$E_n(t) = f^{(n+1)}(\zeta)\frac{\Psi(t)}{(n+1)!}.$$

We now define the function

$$e(x) = f(x) - p_n(x) - \Psi(x)\frac{f(t) - p_n(t)}{\Psi(t)}.$$

Observe that $e(x) = 0$ at the tabulated points x_i and t. Hence, $e(x)$ has $n+2$ zeros in the interval containing the tabulated points. Now apply Rolle's Theorem which states that between any two zeros of $e(x)$ there must be at least one zero of $e'(x)$. Thus, the first derivative $e'(x)$ has at least $n+1$ zeros. Similarly $e''(x)$ has at least n zeros. Continuing in this way, we finally find that $e^{(n+1)}(x)$ has at least one zero.

Let such a zero be z. Then

$$e^{(n+1)}(z) = f^{(n+1)}(z) - p_n^{(n+1)}(z) - \Psi^{(n+1)}(z)\frac{f(t) - p_n(t)}{\Psi(t)} = 0$$

But $p_n^{(n+1)}(z) = 0$ since the degree of $p_n(x) \leq n$. Also $\Psi^{(n+1)}(z) = (n+1)!$ (see Exercise 1). Hence, we have

$$f^{(n+1)}(z) - (n+1)!\frac{f(t) - p_n(t)}{\Psi(t)} = 0$$

Finally, solve for $f(t) - p_n(t)$ to get (5.17).

$$E_n(t) = f(t) - p_n(t) = f^{(n+1)}(z)\frac{\Psi(t)}{(n+1)!}$$

The next result addresses the special case when $f(x)$ is defined on $[a, b]$, which contains equally spaced nodes.

Assume $f(x)$ is defined on $[a, b]$, which contains equally spaced nodes $x_k = x_0 + hk$, $k = 0, 1, \ldots, n$. In addition, assume that $f^{(n+1)}$ is continuous on $[a, b]$ and satisfies $f^{(n+1)}(x) \leq M$. Then on $[a, b]$

$$|f(x) - p_n(x)| \leq \frac{1}{4(n+1)}M\left(\frac{b-a}{n}\right)^{n+1}$$

The proof can be obtained from the previous result. It can be shown that (see Exercise 5)

$$|\Psi(x)| \leq \frac{1}{4} n!\, h^{n+1}, \quad h = \frac{b-a}{n} \quad \forall x \in [a, b].$$

Therefore, from Eqn. (5.17) we get

$$|f(x) - p_n(x)| \leq \frac{1}{(n+1)!} f^{(n+1)}(\zeta) \frac{1}{4} h^{(n+1)} n! \leq \frac{1}{4(n+1)} M \left(\frac{b-a}{n}\right)^{n+1}.$$

EXAMPLE 5.5

An interpolating polynomial is to be used to approximate $f(x) = \cos x$ with 11 equally spaced nodes in $[1, 2]$. What bound can be placed on the error?

We have,
$n = 10$, $a = 1$, $b = 2$, $f^{(10)}(x) = -\cos x$ and $|f^{(10)}(x)| \leq 1 = M$.
Thus,

$$|p(x) - \cos x| \leq \frac{1}{4(11)} \left(\frac{2-1}{10}\right)^{11} \approx 2.27 \times 10^{-13}$$

EXERCISE SET 5.3

1. Show that the $(n+1)$th derivative of $\Psi(x)$ defined by Eqn. (5.18) is

$$\Psi^{(n+1)}(x) = (n+1)!$$

2. Show that the third order divided difference $f[x_0, x_1, x_2]$ of f is invariant under all permutations of x_0, x_1, x_2.

3. Let p_n be a polynomial of degree n. Show that the divided difference of order k of p_n is zero if $k > n$.

4. Show that

$$f[x_0, x_1, x_2] = \frac{\begin{vmatrix} 1 & x_0 & f_0 \\ 1 & x_1 & f_1 \\ 1 & x_2 & f_2 \end{vmatrix}}{\begin{vmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{vmatrix}}$$

5. In interpolating with $(n+1)$ equally spaced nodes on an interval $[a, b]$, show that for any $x \in [a, b]$

$$\prod_{i=0}^{n} |x - x_i| \leq \frac{1}{4} h^{(n+1)} n! \quad \text{with} \quad h = \frac{b-a}{n}$$

5.4 LAGRANGE INTERPOLATING POLYNOMIAL

There are many forms of $p_n(x)$ that are different from the Newton form. Among them is the form known by the name of **Lagrange**, which we will derive in this section. As before, we seek a polynomial of degree n which passes through the $n+1$ points given by Table 5.2. To proceed, let us first define the functions which have the property

$$L_j(x_i) = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ if } i \neq j \end{cases} \tag{5.19}$$

These functions are known as **cardinal functions**.

We now write the nth Lagrange interpolating polynomial in the form

$$p_n(x) = \sum_{i=0}^{n} f(x_i) L_i(x). \tag{5.20}$$

Observe that $p_n(x)$ easily satisfies property (5.1). So, if we define

$$\begin{aligned} \mathrm{L}_i(x) &= \frac{(x - x_0)(x - x_1) \cdots (x - x_{i-1})(x - x_{i+1}) \cdots (x - x_n)}{(x_i - x_0)(x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)} \\ &= \prod_{\substack{j=0 \\ j \neq i}}^{n} \left[\frac{x - x_j}{x_i - x_j} \right] \quad i = 0, 1, \ldots, n \end{aligned} \tag{5.21}$$

then the degree of $p_n(x) \leq n$ since all $L_i(x)$ have degree n.

Notice that $L_i(x)$ satisfies property (5.19) since at $x = x_i$ the numerator and denominator of Eqn. (5.21) will be equal. So $L_i(x_i) = 1$. Further, if $x = x_j$ and $i \neq j$, then the numerator of the (5.21) will be zero, and $L_i(x_j) = 0$.

The expression (5.20) is called the **Lagrange form** of $p_n(x)$.

EXAMPLE 5.6

Derive the Lagrange polynomial that interpolates the data in the following table.

x	-1	0	3
$f(x)$	8	-2	4

From Eqn. (5.21) it follows

$$L_0(x) = \frac{(x-0)(x-3)}{(-1-0)(-1-3)} = \frac{1}{4}x(x-3)$$

$$L_1(x) = \frac{(x+1)(x-3)}{(0+1)(0-3)} = -\frac{1}{3}(x+1)(x-3)$$

$$L_2(x) = \frac{(x+1)(x-0)}{(3+1)(3-0)} = \frac{1}{12}x(x+1)$$

Thus, from Eqn. (5.20) the Lagrange interpolating polynomial is

$$p_2(x) = 2x(x-3) + \frac{2}{3}(x+1)(x-3) + \frac{1}{3}x(x+1)$$

As one observes from this example, one of the disadvantages of the Lagrange polynomial method is that it requires many computations. Another disadvantage of the Lagrange polynomial method is that if one wants to add or subtract a point from the table used to construct the polynomial, one has to start with a complete new set of calculations.

EXAMPLE 5.7

Determine the coefficients of the Lagrange interpolating polynomial that interpolates the data in Example 5.6 and evaluate it at $x = -0.5$.

m-function 5.2

The following MATLAB function **lagrange.m** determines the coefficients of the Lagrange interpolating polynomial *p(x)* and computes *p(a)*. INPUTS are the vectors *x* and *y* and the value of *a*.

```
function lagrange(x,y,a)
% Coefficients of the Lagrange interpolating polynomial.
n=length(x);
p=0;
for k=1:n
  b(k)=1;
  d(k)=1;
  for j=1:n
    if j~= k
      b(k)=b(k)*(x(k)-x(j));
      d(k)=d(k)*(a-x(j));
    end
  end
  c(k)=y(k)/b(k);
  p=p+c(k)*d(k);
end
c
fprintf('\n   p(a)= %10.6f',p)
fprintf('\n')
```

Using the MATLAB function lagrange, we get the output shown in Table 5.7.

EXERCISE SET 5.4

1. Derive the Lagrange polynomial that interpolates the data in the following table.

x	0	2	4	6
y	1	-1	3	4

2. Derive the cardinal functions that can be used to interpolate the following table.

```
» x=[-1 0 3]';
» y=[8 -2 4]';
» lagrange(x,y,-0.5)

c =

    2.0000   0.6667   0.3333

   p(a)=  2.250000
```

Table 5.7 Coefficients of Lagrange interpolating polynomial for Example 5.7.

x	-1	0.5	1
y	3	1	4

3. Let $f(x) = 2x^2e^x + 1$. Construct a Lagrange polynomial of degree two or less using $x_0 = 0, x_1 = 0.5$, and $x_2 = 1$. Approximate $f(0.8)$.

APPLIED PROBLEMS FOR CHAPTER 5

1. S.H.P. Chen and S.C. Saxena experimental data for the emittance e of tungsten as a function of the temperature T (*Ind. Eng. Chem. Fund.*, **12**, 220 (1973)) are given by the following table.

$T,^0K$	300	400	500	600	700	800	900	1000	1100
e	0.024	0.035	0.046	0.056	0.067	0.083	0.097	0.111	0.125
$T,^0K$	1200	1300	1400	1500	1600	1700	1800	1900	2000
e	0.140	0.155	0.170	0.186	0.202	0.219	0.235	0.252	0.269

They found that the equation

$$e(T) = 0.02424\left(\frac{T}{303.16}\right)^{1.27591}$$

correlated the data accurately to three digits. Find the coefficients for Newton's interpolating polynomial $p(T)$ that interpolates these data. Compare the values of $p(T)$ with the values of $e(T)$ at the points midway between the tabulated temperatures. Plot $p(T)$ and $e(T)$ as functions of T in the interval $[300, 2000]$.

2. The following table shows the values of the thermal conductivity, k (BTU/hr ft ^oF), of carbon dioxide gas and the values of the viscosity, μ (lb/ft hr), of liquid ethylene glycot, at various temperature $T(^oF)$.

T	31	211	391	571	
k	0.0084	0.0132	0.0180	0.0227	
T	0	50	100	150	200
μ	241	82.2	30.3	12.5	5.54

In each case, determine the simplest interpolating polynomial that is likely to predict k and μ in the specified ranges of temperatures.

3. When the initial value problem, $dy/dt = f(t, y)$, subject to an initial condition is solved numerically, $y(t)$ is calculated at the mesh points of a given interval (see Chapter 9). When a certain problem is solved, the values given in the following table were obtained.

t	0.0	0.1	0.2	0.3	0.4	0.5
y	-1.0	-0.90	-0.79	-0.67	-0.54	-0.41

The numerical method does not give the values of y at points other than the mesh points. Use an interpolating polynomial to estimate y at $t = 0.26$, 0.35, and 0.43.

4. Consider the graph of the function

$$f(x) = \frac{1}{1 + x^2}$$

(a) Determine the polynomial $p_{10}(x)$ of 10th degree that interpolates the points with abscissa $-5, -4, \ldots, 4, 5$, all spaced one unit apart.

(b) Plot both p and f in the interval $[-5, 5]$. Show that the wiggles in $p_{10}(x)$ occur near the ends of the interval over which the interval is used by finding the error $E = |f(x) - P_{10}(x)|$ at $x = 4.2, 4.4, 4.6$, and 4.8. This is a characteristic behavior of interpolating polynomials of high degree.

Chapter 6

Interpolation with Spline Functions

Many scientific and engineering phenomena being measured undergo a transition from one physical domain to another. Data obtained from these measurements are better represented by a set of piecewise continuous curves rather than by a single curve. One of the difficulties with polynomial interpolation is that in some cases the oscillatory nature of high-degree polynomials can induce large fluctuations over the entire range when approximating a set of data points. One way of solving this problem is to divide the interval into a set of subintervals and construct a lower-degree approximating polynomial on each subinterval. This type of approximation is called **piecewise polynomial interpolation**.

Piecewise polynomial functions, especially spline functions, have become increasingly popular. Most of the interest has centered on cubic splines because of the ease of their applications to a variety of fields such as, solution of boundary value problems for differential equations and the method of finite elements for the numerical solution of partial differential equations.

In this chapter we shall discuss several types of piecewise polynomials for interpolating a given set of data. The simplest of these is piecewise linear interpolation and the most common one is cubic spline interpolation.

6.1 PIECEWISE LINEAR INTERPOLATION

We start out with the general definition of spline functions. Let f be a real-valued function defined on some interval $[a, b]$ and let the set of

x	$a = x_1$	x_2	$\ldots$	$x_n = b$
y	$f(x_1)$	$f(x_2)$	$\ldots$	$f(x_n)$

Table 6.1 Table of values of a function f.

data points in Table 6.1 be given. For simplicity, assume that

$$a = x_1 < x_2 < \ldots < x_n = b.$$

We have the definition:

DEFINITION 6.1 *A function S is called a* **spline of degree** k *if it satisfies the following conditions:*

1. *S is defined in the interval $[a, b]$.*
2. *$S^{(r)}$ is continuous on $[a, b]$ for $0 \leq r \leq k - 1$.*
3. *S is a polynomial of degree $\leq k$ on each subinterval $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n - 1$.*

Observe that in contrast to polynomial interpolation, the degree of the spline does not increase with the number of points. Here the degree is fixed and one uses more polynomials instead. A simple and familiar example of a piecewise polynomial approximation is piecewise linear interpolation, which consists of connecting a set of data points in Table 6.1 by a series of straight lines as shown in Figure 6.1.

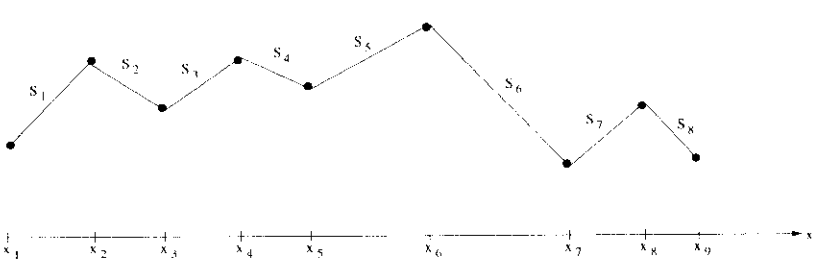


FIGURE 6.1
Piecewise linear interpolation.

This procedure can be described as follows: Let $f(x)$ be a real-valued function defined on some interval $[a, b]$. We wish to construct a piecewise linear polynomial function $S(x)$, which interpolates $f(x)$ at the data points given by Table 6.1, where

$$a = x_1 < x_2 < \ldots < x_n = b.$$

Using the formula of the equation of the line, it is easy to see that the function $S(x)$ is defined by

$$\begin{aligned} S_i(x) &= f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i), \quad i = 1, \ldots, n-1 \\ &= f(x_i) + f[x_{i+1}, x_i](x - x_i) \end{aligned} \tag{6.1}$$

on each subinterval $[x_i, x_{i+1}]$. Outside the interval $[a, b]$, $S(x)$ is usually defined by

$$S(x) = \begin{cases} S_1(x) & \text{if } x < a \\ S_{n-1}(x) & \text{if } x > b \end{cases}$$

The points $x_2, x_3, \ldots, x_{n-1}$, where $S(x)$ changes from one polynomial to another, are called the **breakpoints** or **knots**. Because $S(x)$ is continuous on $[a, b]$, it is called a **spline of degree 1**.

EXAMPLE 6.1

Find a first degree spline interpolating the following table:

x	1	1.5	2	2.5	3
$f(x)$	1	3	7	10	15

Use the resulting spline to approximate $f(2.2)$.

From Eqn. (6.1), we have

$$S_1(x) = \frac{f(1.5) - f(1)}{1.5 - 1}(x - 1) + f(1) = \frac{3 - 1}{0.5}(x - 1) + 1 = 4x - 3$$

$$S_2(x) = \frac{f(2) - f(1.5)}{2 - 1.5}(x - 1.5) + f(1.5) = \frac{7 - 3}{0.5}(x - 1.5) + 3 = 8x - 9$$

$$S_3(x) = \frac{f(2.5) - f(2)}{2.5 - 2}(x - 2) + f(2) = \frac{10 - 7}{0.5}(x - 2) + 7 = 6x - 5$$

$$S_4(x) = \frac{f(3) - f(2.5)}{3 - 2.5}(x - 2.5) + f(2.5) = \frac{15 - 10}{0.5}(x - 2.5) + 10 = 10x - 15$$

m-function 6.1

The following MATLAB function **spl1.m** finds a first-degree spline that interpolates a table of values. INPUTS are a table of function values *x* and *y*.

```
function spl1(x,y,c)
% First-degree spline.
n=length(x);
for i=n-1:-1:2
  dis=c-x(i);
  if(dis>=0)
    break
  end
end
if(dis<0)
 i=1;
 dis=c-x(1);
end
m=(y(i+1)-y(i))/(x(i+1)-x(i));
spl1=y(i)+m*dis;
disp('              linear spline')
disp('_____________________________________________________')
disp(['          x                  y                  c = ',num2str(c),'    '])
disp('_____________________________________________________')
for j=1:n
  fprintf('%12.6f  %12.6f ',x(j),y(j))
  if (j==i)
    fprintf('\n                          S(c)= %10.6f',spl1)
  end;
  fprintf('\n')
end
```

Hence,

$$S(x) = \begin{cases} 4x - 3 & \text{if } x \in [1, 1.5] \\ 8x - 9 & \text{if } x \in [1.5, 2] \\ 6x - 5 & \text{if } x \in [2, 2.5] \\ 10x - 15 & \text{if } x \in [2.5, 3] \end{cases}$$

The value $x = 2.2$ lies in $[2, 2.5]$ and so $f(2.2) \approx 6(2.2) - 5 = 8.2$. Table 6.2 shows the results obtained by using the MATLAB function spl1.

```
x=[1 1.5 2 2.5 3]';
» y=[1 3 7 10 15]';
» spl1(x,y,2.2)
                    linear spline
-----------------------------------------------
   x        y                  c= 2.2
-----------------------------------------------
   1        1
   1.5      3
   2        7
                          S(c)=  8.200000
   2.5      10
   3        15
```

Table 6.2 First degree spline for Example 6.1.

EXERCISE SET 6.1

1. Determine whether the following functions are first degree splines:

   (a)

$$f(x) = \begin{cases} 2x - 1 & \text{if } x \in [-1, 1] \\ -x + 2 & \text{if } x \in [1, 2] \\ 5x & \text{if } x \in [2, 3] \end{cases}$$

   (b)

$$f(x) = \begin{cases} 3x + 5 & \text{if } x \in [0, 1] \\ 2x + 6 & \text{if } x \in [1, 4] \\ x + 10 & \text{if } x \in [4, 5] \end{cases}$$

2. Given the table

x	1	2	3	4	5
$S(x)$	3	4	3	9	1

   find the first degree spline that interpolates the table. Compute $S(2.3)$.

3. Construct a first degree-spline to approximate the function $f(x) = xe^x$ at $x = 0.4$, using the data

x	0.1	0.3	0.5	0.7
$f(x)$	.110517	.404957	.824360	1.409627

4. Construct a first degree spline $S(x)$ to approximate the function $f(x) = e^{\frac{2x}{3}}$ by using the values given by $f(x)$ at $x = 0, 0.02$, and 0.04. Find an approximation for $\int_0^{0.04} e^{\frac{2x}{3}} dx$ by evaluating $\int_0^{0.04} S(x)dx$. Compare the result to the exact value 0.04053811.

5. **Computer assignment:**
Write a computer program in FORTRAN or C to find the first degree spline $S(x)$ that interpolates a set of data points (x_i, y_i) and compute $S(x)$ at a given value α.
Input data to the program should be

   (a) The set of data points (x_i, y_i).

   (b) The given value α.

   Output should consist of:

   (a) The value of $S(x)$ at $x = \alpha$.

6.2 QUADRATIC SPLINE

In many cases, linear piecewise polynomials are unsatisfactory when being used to interpolate the values of a function which deviate considerably from a linear function. In such cases, piecewise polynomials of higher degree are more suitable to use to approximate the function. In this section, we shall discuss the simplest type of differentiable, piecewise polynomial functions, known as **quadratic splines**. As before, consider the subdivision

$$a = x_1 < x_2 < \ldots < x_n = b$$

where $x_1, \ldots, x_n$ are given in Table 6.1. For piecewise linear interpolation, we choose two points $(x_i, f(x_i))$ and $(x_{i+1}, f(x_{i+1}))$ in the

subinterval $[x_i, x_{i+1}]$ and draw a line through those two points to interpolate the data. This approach is easily extended to construct the quadratic splines. Instead of choosing two points, we choose three points in the subinterval $[x_i, x_{i+1}]$ and pass a second degree polynomial through these points as shown in Figure 6.2. We shall show that there is only one such polynomial.

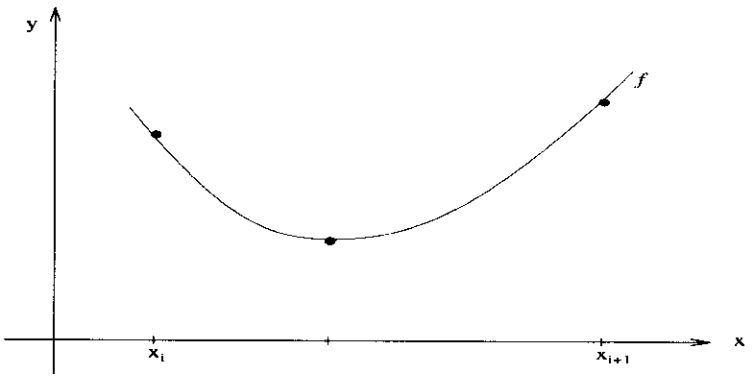


FIGURE 6.2
Quadratic spline.

To construct a quadratic spline $Q(x)$, we first define a quadratic function in each subinterval $[x_i, x_{i+1}]$ by

$$q_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \tag{6.2}$$

where a_i, b_i and c_i are constants to be determined.

Now by definition 6.1, $Q(x)$ must satisfy the conditions

$$Q(x) = q_i(x) \quad \text{on } [x_i, x_{i+1}] \quad \text{for } i = 1, 2, \ldots, n-1. \tag{6.3}$$

$$q_i(x_i) = f(x_i) \tag{6.4}$$

$$q_i(x_{i+1}) = f(x_{i+1}) \tag{6.5}$$

$Q'(x)$ is continuous on $[a, b]$ if

$$q_i'(x_i) = d_i \quad \text{and} \quad q_i'(x_{i+1}) = d_{i+1}. \tag{6.6}$$

Here the values of d_i will be defined later. Using conditions (6.4) and (6.6), it is easy to see that $q_i(x)$ is uniquely defined on $[x_i, x_{i+1}]$ (see Exercise 1) by

$$q_i(x) = f(x_i) + d_i(x - x_i) + \frac{d_{i+1} - d_i}{2(x_{i+1} - x_i)}(x - x_i)^2. \tag{6.7}$$

We now use condition (6.5) to obtain d_i from the recursive formula

$$d_{i+1} = -d_i + 2\left[\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}\right], \qquad i = 1, 2, \ldots, n-1 \tag{6.8}$$

with d_1 arbitrary.

Thus, given the data in Table 6.1 and an arbitrary value for d_1, the quadratic spline $Q(x)$ is uniquely determined by formulas (6.3), (6.7), and (6.8).

EXAMPLE 6.2

Find a quadratic spline interpolating the following table.

x	-2	-1.5	0	1.5
$f(x)$	5	3	1	2

Let $d_1 = 0$, then from (6.8) we have

$$d_2 = 2\left[\frac{3-5}{-1.5+2}\right] = -8$$

$$d_3 = 8 + 2\left[\frac{1-3}{0+1.5}\right] = 16/3$$

$$d_4 = -\frac{16}{3} + 2\left[\frac{2-1}{1.5-0}\right] = -4$$

We now use (6.7) to get a quadratic spline $Q(x)$ defined by

$$q_1(x) = 5 - 8(x+2)^2 \quad \text{in} \quad [-2, -1.5]$$

$$q_2(x) = 3 - 8(x+1.5) + \frac{40}{9}(x+1.5)^2 \quad \text{in} \quad [-1.5, 0]$$

$$q_3(x) = 1 + \frac{16}{3}x - \frac{28}{9}x^2 \quad \text{in} \quad [0, 1.5]$$

EXERCISE SET 6.2

1. Show formula (6.7) using (6.2), (6.4), and (6.6).

2. The following data are taken from a polynomial.

x	-1	0	1	2
$p(x)$	-4	-1	2	23

Find the quadratic spline $Q(x)$ that interpolates this data. Compute $Q(1.5)$.

3. Determine the coefficient a and b so that the following function is a quadratic spline.

$$f(x) = \begin{cases} -x+1 & \text{if } 0 \le x \le 1 \\ ax^2+b & \text{if } 1 \le x \le 2 \end{cases}$$

4. Determine the coefficient a, b, c, and d so that the following function is a quadratic spline.

$$f(x) = \begin{cases} 2x & \text{if } -1 \le x \le 1 \\ ax^2+bx+c & \text{if } 1 \le x \le 2 \\ -dx & \text{if } 2 \le x \le 4 \end{cases}$$

5. Determine whether the following function is a quadratic spline:

$$f(x) = \begin{cases} 2x^2-1 & \text{if } -2 \le x \le -1 \\ x^2+x+1 & \text{if } -1 \le x \le 0 \\ 3x+1 & \text{if } 0 \le x \le 1 \end{cases}$$

6. Show that the following function is a quadratic spline.

$$f(x) = \begin{cases} 2x^2 & \text{if } -1 \le x \le 1 \\ x^2+2x-1 & \text{if } 1 \le x \le 3 \\ 8x-10 & \text{if } 3 \le x \le 4 \end{cases}$$

6.3 NATURAL CUBIC SPLINES

Cubic spline functions are the most popular spline functions because they are a very satisfactory device for smooth interpolation and numerical differentiation. The difficulty with quadratic splines is that

their curvature is discontinuous at each breakpoint. So, if we have a smooth function to interpolate, cubic splines are more desirable to use and adequately provide accurate interpolation.

Again, we start with the Table 6.1.
We now construct the cubic spline $S(x)$ subject to the conditions given by definition 6.1 in Section 6.1.

In each subinterval $[x_i, x_{i+1}]$, we write $S(x)$ in the form

$$\begin{aligned} S_i(x) &= a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \\ &= S(x), \quad i = 1, 2, \ldots, n-1 \end{aligned} \tag{6.9}$$

where a_i, b_i, c_i, and d_i are constant to be determined. For notational convenience, let

$$h_i = x_{i+1} - x_i.$$

In order to fit the data, $S(x)$ must satisfy

$$f(x_i) = S(x_i)$$

so

$$a_i = f(x_i) \quad i = 1, 2, \ldots, n$$

and if the condition on the continuity of $S(x)$ is applied,

$$\begin{aligned} S_{i+1}(x_{i+1}) &= S_i(x_{i+1}) \\ f(x_{i+1}) &= a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3 \\ a_{i+1} &= a_i + b_i h_i + c_i h_i^2 + d_i h_i^3, \quad i = 1, 2, \ldots, n-1 \end{aligned} \tag{6.10}$$

By differentiating Eqn. (6.9), we obtain

$$\begin{aligned} S_i'(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \\ S_i''(x) &= 2c_i + 6d_i(x - x_i) \end{aligned}$$

The continuity condition on the first derivative implies

$$\begin{aligned} S_i'(x_i) &= S_{i-1}'(x_i) \\ b_i &= b_{i-1} + 2c_{i-1}(x_i - x_{i-1}) + 3d_{i-1}(x_i - x_{i-1})^2 \\ &= b_{i-1} + 2c_{i-1}h_{i-1} + 3d_{i-1}h_{i-1}^2, \quad i = 2, 3, \ldots, n \end{aligned} \tag{6.11}$$

Similarly, imposing the continuity condition on the second derivative gives

$$\begin{aligned} S_i''(x_i) &= S_{i-1}''(x_i) \\ 2c_i &= 2c_{i-1} + 6d_{i-1}(x_i - x_{i-1}) \\ c_i &= c_{i-1} + 3d_{i-1}h_{i-1}, \quad i = 2, 3, \ldots, n \end{aligned} \tag{6.12}$$

By increasing the index i by one, Eqn. (6.12) can be also written in the form

$$c_{i+1} = c_i + 3d_i h_i \tag{6.13}$$

for $i = 1, 2, \ldots, n-1$.

Now, by solving Eqn. (6.13) for d_i and substituting its value into Eqn. (6.10), we get, after rearranging the equation,

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} + 2c_i), \quad i = 1, 2, \ldots, n-1 \tag{6.14}$$

or

$$b_{i-1} = \frac{1}{h_{i-1}}(a_i - a_{i-1}) - \frac{h_{i-1}}{3}(c_i + 2c_{i-1}), \quad i = 2, 3, \ldots, n \tag{6.15}$$

Similarly, solve Eqn. (6.12) for d_{i-1} and substitute its value into Eqn. (6.11) to get

$$b_i = b_{i-1} + h_{i-1}(c_i + c_{i-1}), \quad i = 2, 3, \ldots, n \tag{6.16}$$

Finally, by combining Eqns. (6.14), (6.15), and (6.16) we get

$$h_{i-1}c_{i-1} + u_i c_i + h_i c_{i+1} = v_i, \quad i = 2, 3, \ldots, n-1 \tag{6.17}$$

where

$$u_i = 2(h_{i-1} + h_i), \;\; v_i = 3w_i - 3w_{i-1} \quad \text{and} \quad w_i = \frac{1}{h_i}(a_{i+1} - a_i)$$

Eqn. (6.17) is a linear system in n unknowns and $n-2$ equations. To solve any system of equations numerically, it is necessary that the number of equations be exactly equal to the number of unknowns. This indicates that two more conditions are required.

There are several possible choices for these conditions on the second derivatives evaluated at the endpoints, which provide that additional conditions to determine all the unknowns. The simplest choice is given by the natural boundary

$$S''(x_1) = S''(x_n) = 0 \tag{6.18}$$

which leads to a spline function known as a **natural cubic spline** whose graph is obtained by forcing a long flexible rod to go through

each of the data points.
So, if condition (6.18) is applied to $S''(x)$, we obtain two new equations,

$$c_1 = 0 \quad \text{and} \quad c_n = 0. \tag{6.19}$$

The two Eqns. (6.19) together with the Eqn. (6.17) produce a linear, tridiagonal system of the form

$$A\mathbf{x} = \mathbf{b}$$

where

$$A = \begin{bmatrix} 1 & 0 & & & & \\ h_1 & 2(h_1 + h_2) & h_2 & & 0 & \\ & & \ddots & & & \\ & 0 & h_{n-2} & 2(h_{n-2} + h_{n-1}) & h_{n-1} \\ & & & 0 & 1 \end{bmatrix}$$

$$\mathbf{b} = \begin{bmatrix} 0 \\ 3(a_3 - a_2)/h_2 - 3(a_2 - a_1)/h_1 \\ \vdots \\ 3(a_n - a_{n-1})/h_{n-1} - 3(a_{n-1} - a_{n-2})/h_{n-2} \\ 0 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix}$$

This system is strictly diagonally dominant, so an elimination method can be used to find the solution without the need for pivoting. Having obtained the values of $\{c_i\}_{i=1}^n$, the remainder of the spline coefficients for

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

are obtained using the formulas

$$\begin{aligned} a_i &= f(x_i) \\ b_i &= \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(c_{i+1} + 2c_i) \\ d_i &= \frac{c_{i+1} - c_i}{3h_i} \end{aligned} \tag{6.20}$$

for $i = 1, 2, \ldots n - 1$.

m-function 6.2

The following MATLAB function **spl3.m** finds a natural cubic spline that interpolates a table of values. INPUTS are a table of function values *x* and *y*; the number of intermediate points *m* at which *S(x)* is to be approximated.

```
function function spl3(x,y,m)
% Cubic spline
n=length(x);
for i=1:n-1
  h(i)=x(i+1)-x(i);
  b(i)=(y(i+1)-y(i))/h(i);
end
u(2)=2*(h(1)+h(2));
v(2)=6*(b(2)-b(1));
for i=3:n-1
  u(i)=2*(h(i-1)+h(i))-h(i-1)^2/u(i-1);
  v(i)=6*(b(i)-b(i-1))-h(i-1)*v(i-1)/u(i-1);
end
z(n)=0;
for i=n-1:-1:2
  z(i)=(v(i)-h(i)*z(i+1))/u(i);
end
z(1)=0;
disp('              cubic spline')
disp('_____________________________________________________')
disp('              x                    S(x)                 ')
disp('_____________________________________________________')
for i=1:n-1
  for j=1:m+1
    r=(x(i+1)-x(i))/(m+1);
    t=x(i)+(j-1)*r;
    dis=(j-1)*r;
    hh=x(i+1)-x(i);
    bb=(y(i+1)-y(i))/hh-hh*(z(i+1)+2*z(i))/6;
    q=0.5*z(i)+dis*(z(i+1)-z(i))/(6*hh);
    q=bb+dis*q;
    spl3=y(i)+dis*q;
    fprintf('%12.5f %17.5f \n',t,spl3)
  end
end
fprintf('%12.5f %17.5f \n',x(n),y(n))
fprintf('\n')
```

EXAMPLE 6.3

Use the values given by $f(x) = x^3 + 2$ at $x = 0, 0.2, 0.4, 0.6, 0.8$, and 1.0 to find an approximation of $f(x)$ at $x = 0.1, 0.3, 0.5, 0.7$, and 0.9 using natural cubic spline interpolation.

We have the table

x	0.0	0.2	0.4	0.6	0.8	1.0
$f(x)$	2.0	2.008	2.064	2.216	2.512	3.0

From Eqn. (6.17), we have

$$u_1 = 0.8,\ u_2 = 0.8,\ u_3 = 0.8,\ u_4 = 0.8$$
$$v_1 = 0.72,\ v_2 = 1.44\ ,\ v_3 = 2.16,\ v_4 = 2.88$$

Using these values, we obtain the linear system of equations

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0.2 & 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0.2 & 0.8 & 0.2 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 0.2 & 0.8 & 0.2 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.72 \\ 1.44 \\ 2.16 \\ 2.88 \\ 0 \end{bmatrix}.$$

The solution of this system is

$$\mathbf{c} = [0.0, 0.586, 1.257, 1.585, 3.204, 0.0]^T.$$

Using Eqn. (6.20), we obtain the coefficients

$$\mathbf{a} = \begin{pmatrix} 2 \\ 2.008 \\ 2.064 \\ 2.216 \\ 2.512 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 0.001 \\ 0.118 \\ 0.487 \\ 1.055 \\ 2.013 \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} 0.976 \\ 1.120 \\ 0.545 \\ 2.699 \\ -5.340 \end{pmatrix}$$

Hence,

$$S(x) = \begin{cases} S_1(x),\ x \in [0.0, 0.2] \\ S_2(x),\ x \in [0.2, 0.4] \\ S_3(x),\ x \in [0.4, 0.6] \\ S_4(x),\ x \in [0.6, 0.8] \\ S_5(x),\ x \in [0.8, 1.0] \end{cases}$$

with

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = 1, \ldots, 5.$$

For example, the value $x = 0.5$ lies in the interval $[0.4, 0.6]$, so

$$\begin{aligned} S_3(0.5) &= 2.064 + 0.487(0.5 - 0.4) + 1.257(0.5 - 0.4)^2 + 0.545(0.5 - 0.4)^3 \\ &\approx 2.126. \end{aligned}$$

A summary of the calculations is given in Table 6.3.

```
» x=[0:0.2:1]';
» y=x.^3+2;
» spl3(x,y,2)
```

cubic spline

x	S(x)	Error
0	2	0
0.1	2.00107	0.000072
0.2	2.008	0
0.3	2.02678	0.000215
0.4	2.064	0
0.5	2.12579	0.000789
0.6	2.216	0
0.7	2.34006	0.002943
0.8	2.512	0
0.9	2.73998	0.010981
1	3	0

Table 6.3 Natural cubic spline for Example 6.3.

EXAMPLE 6.4

Given the table

x	0	4	6	7	13	15	19	24	26
$f(x)$	0	3	8	12	7	5	2	3	5

use a cubic spline to approximate this data, and use the spline to approximate $f(x)$ at $x = 17$.

For this example, we choose $n = 3$ in order to approximate $f(17)$. A summary of the calculated values is shown in Table 6.4, and Figure 6.3 shows the graph of the corresponding cubic spline.

```
» x=[0 4 6 7 13 15 19 24 26]';
» y=[0 3 8 12 7 5 2 3 5]';
» spl3(x,y,3)
```

cubic spline

x	S(x)	x	S(x)
0	0	13.5	6.31667
1	0.38404	14	5.79912
2	0.91446	14.5	5.38201
3	1.73765	15	5
4	3	16	4.18356
4.5	3.85054	17	3.34531
5	4.91942	18	2.58441
5.5	6.27859	19	2
6	8	20.25	1.63444
6.25	9.00626	21.5	1.6874
6.5	10.05343	22.75	2.14665
6.75	11.07389	24	3
7	12	24.5	3.44654
8.5	14.55232	25	3.9389
10	13.25118	25.5	4.46181
11.5	10.07445	26	5
13	7		

Table 6.4 Natural cubic spline for Example 6.4.

EXAMPLE 6.5

Determine whether the following function is a cubic spline.

$$f(x) = \begin{cases} S_1(x) = -x^3 - 3x & \text{if } 0 \le x \le 1 \\ S_2(x) = x^3 - 6x^2 + 3x - 2 & \text{if } 1 \le x \le 2 \end{cases}$$

Since both $S_1(x)$ and $S_2(x)$ are polynomials of degree three, the only

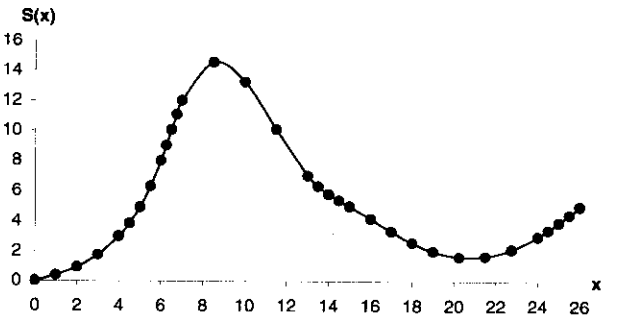


FIGURE 6.3
$S(x)$ from Example 6.4.

condition we need to check is the continuity of $S(x)$, $S'(x)$, and $S''(x)$ at $x = 1$. We have:

1. $S(x)$ must satisfy the continuity condition at $x = 1$.
   $S_1(1) = -4 = S_2(1)$.

2. $S'(x)$ must satisfy the continuity condition at $x = 1$.
   $S_1'(x) = -3x^2 - 3$ and $S_1'(1) = -6$
   $S_2'(x) = 3x^2 - 12x + 3$ and $S_2'(1) = -6$

3. $S''(x)$ must satisfy the continuity condition at $x = 1$.
   $S_1''(x) = -6x$ and $S_1''(1) = -6$
   $S_2''(x) = 6x - 12$ and $S_2''(1) = -6$

Thus, $S(x)$ is a cubic spline on $[0, 2]$.

A practical feature of cubic splines is that they minimize the oscillations in the fit between the interpolating points. This is unlike all functions, $f(x)$, which are twice differentiable on $[a, b]$ and interpolate a given set of data points. The next theorem explains this phenomenon.

THEOREM 6.1
If f is twice-continuously differentiable on $[a, b]$ and S is the natural cubic spline that interpolates f at the knots $a = x_1 < x_2 < \cdots < x_n = b$, then

$$\int_a^b [S''(x)]^2 dx \leq \int_a^b [f''(x)]^2 dx.$$

Proof:. The proof is obtained by a series of integration by parts. Let

$$e(x) = f(x) - S(x).$$

We have

$$\begin{aligned}\int_a^b [f''(x)]^2 dx &= \int_a^b [S''(x) + e''(x)]^2 dx \\ &= \int_a^b [S''(x)]^2 dx + \int_a^b [e''(x)]^2 dx + 2\int_a^b e''(x)S''(x)\,dx\end{aligned} \tag{6.21}$$

By integrating the last integral on the right-hand side, by parts, we obtain

$$\int_a^b e''(x)S''(x)\,dx = S''(x)e'(x)\Big]_a^b - \int_a^b e'(x)S'''(x)\,dx. \tag{6.22}$$

Since $S''(a) = S''(b) = 0$, we have

$$S''(x)e'(x)\Big]_a^b = S''(b)e'(b) - S''(a)e'(a) = 0. \tag{6.23}$$

Moreover, since $S'''(x) = 6d_k =$ constant on the subinterval $[x_k, x_{k+1}]$, it follows that

$$\int_{x_k}^{x_{k+1}} e'(x)S'''(x)\,dx = 6d_k \int_{x_k}^{x_{k+1}} e'(x)\,dx = [6d_k e(x)]_{x_k}^{x_{k+1}} = 0.$$

Here, use has been made of the fact that $e(x_k) = f(x_k) - S(x_k) = 0$ for $k = 1, ..., n$. Hence,

$$\int_a^b e'(x)S'''(x)\,dx = \sum_{k=1}^{n-1} \int_{x_k}^{x_{k+1}} e'(x)S'''(x)\,dx = 0. \tag{6.24}$$

Finally, using Eqns. (6.21), (6.22), (6.23), and (6.24), we get

$$\int_a^b [S''(x)]^2 dx \le \int_a^b [S''(x)]^2 dx + \int_a^b [e''(x)]^2 dx = \int_a^b [f''(x)]^2 dx.$$

EXERCISE SET 6.3

1. Show that

$$f(x) = \begin{cases} -2x^3 & \text{if } 0 \le x \le 1 \\ (x-1)^3 - 6(x-1)^2 - 6(x-1) - 2 & \text{if } 1 \le x \le 3 \end{cases}$$

is a cubic spline function.

2. Given the table

x	1.4	1.8	2.2
$f(x)$	3.12	2.81	1.7

use natural cubic spline interpolation to approximate $f(1.6)$.

3. (Continuation). Compute the integral of the spline over $[1.4, 2.2]$.

4. Determine a, b, c, and d so that the following function is a natural cubic spline.

$$f(x) = \begin{cases} -3x^3 & \text{if } 0 \le x \le 2 \\ a(x-2)^3 + b(x-2)^2 + c(x-2) + d & \text{if } 2 \le x \le 3 \end{cases}$$

5. Construct a natural cubic spline that interpolates the function $f(x) = xe^x$ at $x = 0, 0.25, 0.5, 0.75$, and 1.0. Compute the integral of the spline over $[0, 1]$ and compare the result to the actual value of

$$\int_0^1 xe^x dx = 1$$

6. Determine whether the following function is a cubic spline

$$f(x) = \begin{cases} -x^2 + 1 & \text{if } 0 \le x \le 1 \\ (x-1)^3 + 2(x-1) & \text{if } 1 \le x \le 2 \\ 2x - 1 & \text{if } 2 \le x \le 3 \end{cases}$$

7. Given the table

x	2.0	2.1	2.2	2.3
$f(x)$	1.5	2.0	3.8	5.1

approximate $f(2.05), f(2.15)$, and $f(2.25)$ using natural cubic spline approximation.

8. Construct a natural cubic spline that interpolates the function $f(x) = 1/(1 + x^2)$ at $x = -2, -1, 0, 1$, and 2. Compare the interpolated values with the true values at $x = -1.5, -0.5, 0.5$, and 1.5.

9. Use the MATLAB function **spl3** to interpolate the function $f(x) = \sin x$ at 10 equally spaced points in the interval $[0, \pi]$. Evaluate the spline at points between the data points (midpoints) and compare the values of the spline with the values of $\sin x$.

10. **Computer assignment:**
Write a computer program in FORTRAN or C to find a natural cubic spline $S(x)$ that interpolates a set of data points (x_i, y_i) and evaluate the spline at points between the data points.
Input data to the program should be

    (a) The set of data points (x_i, y_i).

    Output should consist of:

    (a) The value of $S(x)$ between the data points.

    Test your program to solve Exercise 7.

APPLIED PROBLEMS FOR CHAPTER 6

1. Generate a table of values by taking

$$x_i = i - 6, \qquad i = 1, 2, \ldots, 10$$

$$y_i = \frac{1}{1 + x_i^2}$$

Find a natural cubic spline that interpolates this table. Compare the values of the spline with the values of $y = 1/(1 + x^2)$ at points between the data points (midpoints).

2. The U.S. population from 1880 and 1980 is given in Table 6.5.

    (a) Find a natural spline that interpolates this table.

    (b) What does this spline show for the 1975 population?

3. A biologist who is studying the growth of a bacteria culture, recorded the following data

Year	Population
1880	50,155,780
1890	62,947,710
1900	75,994,570
1910	91,972,265
1920	105,710,623
1930	122,775,625
1940	131,669,270
1950	150,697,365
1960	179,323,170
1970	203,235,290
1980	226,504,820

Table 6.5 Population model.

t	0	3	6	9
$p(t)$	2.0	3	4	6

where $p(t)$ denotes the number of bacteria at time t (minutes). Use natural cubic spline to estimate the number of bacteria at $t = 5$.

Chapter 7

The Method of Least Squares

In approximation theory, two general types of problems arise in fitting tabular data. The first one consists of finding an approximating function (perhaps a piecewise polynomial) that passes through every point in the table. The other problem consists of finding the "best" function that can be used to represent the data, but does not exactly pass through every data point. This category of problems is called **curve fitting** and will be the subject of this chapter.

In Chapter 5, we constructed polynomial approximations to tabular data using interpolation methods. Interpolating polynomials are best used for approximating a function f whose values are known with high accuracy. Often, however, the tabulated data are known to be only approximate. More precisely, in most of the situations, data are given by a set of measurements having experimental errors. Hence, interpolation in this case is of little use, if not dangerous. Consider, for example, a simple physical experiment in which a spring is stretched from its equilibrium position by a known external force (see Figure 7.1).

Suppose we consecutively apply forces of $1, 2, 3$, and 4 kilos to the spring and find that its displacements are $3.3, 6.1, 9.8$ and 12.3 centimeters, respectively. Hooke's law states that the force F and the displacement x are related by the linear function $F = kx$, where k is a constant characteristic of the spring, called the *spring constant*. The problem we want to solve is to determine k from the experimental data. The usual approach to this problem is to seek an approximating function that best fits these data points. In the following sections, we will examine the most commonly used technique for solving the above problem, known as the method of **least squares**.

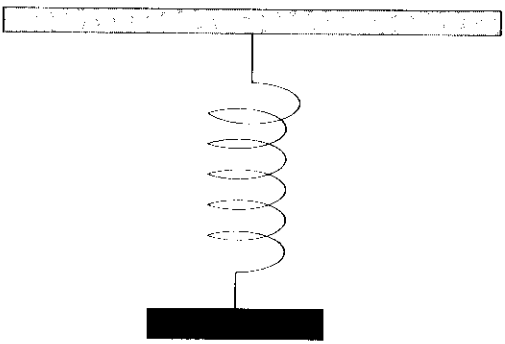


FIGURE 7.1
Spring.

7.1 LINEAR LEAST SQUARES

Consider the experimental data in Table 7.1. The problem of data

x	1	2	3	4	5
y	1	1	2	2	4

Table 7.1 Linear least squares fit.

fitting consists of finding a function f that "best" represents the data which are subject to errors. A reasonable way to approach the problem is to plot the data points in an xy-plane and try to recognize the shape of a *guess function* $f(x)$ (see Figure 7.2) such that

$$f(x) \approx y.$$

The best guess function would be the one that is simple in form and tends to "smooth" the data. Therefore, a reasonable guess function to the data in Figure 7.2 might be a linear one, that is

$$f(x) = ax + b.$$

Having selected a particular function $f(x)$, the problem becomes that of finding the values of the *parameters* a and b that make $f(x)$ the "best" function to fit the data.

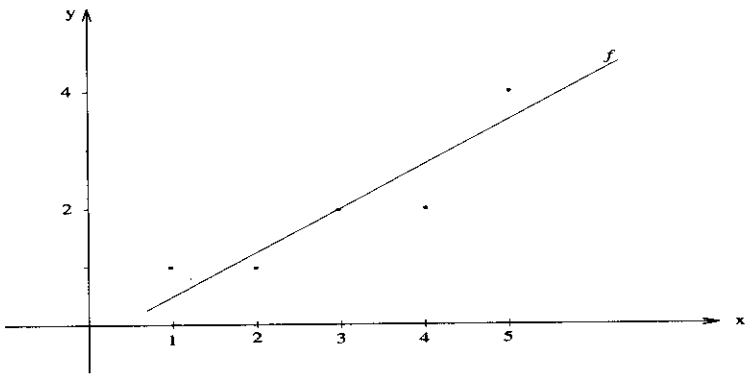


FIGURE 7.2
Least squares line that fits the data in Table 7.1.

Letting $e_i = f(x_i) - y_i$ with $1 \leq i \leq 5$, the least squares criterion consists of minimizing the sum

$$\begin{aligned} E(a,b) &= \sum_{i=1}^{5} (e_i)^2 \\ &= \sum_{i=1}^{5} [f(x_i) - y_i]^2 \\ &= \sum_{i=1}^{5} [(ax_i + b) - y_i]^2 . \end{aligned} \tag{7.1}$$

Here $E(a,b)$ is considered to be a function of two variables. This method of choosing a and b is commonly used in engineering, economics, and many other sciences.

We know from calculus that the minimum of (7.1) will occur when

$$\frac{\partial E(a,b)}{\partial a} = 0 \quad \text{and} \quad \frac{\partial E(a,b)}{\partial b} = 0. \tag{7.2}$$

From Table 7.1, we get, after taking the partial derivatives in (7.2) and rearranging terms:

$$\begin{aligned} 15a + 5b &= 10 \\ 55a + 15b &= 37 \end{aligned}$$

The solution of this linear system of equations is $a = 0.7$ and $b = 0.1$, so the best function in the least squares sense is

$$f(x) = 0.7x - 0.1.$$

x	x_1	x_2	$\cdots$	x_n
y	y_1	y_2	$\cdots$	y_n

Table 7.2 Table of x and y values.

The general problem of approximating the data in Table 7.2 with a linear function $f(x) = ax + b$ using the least squares involves minimizing

$$E(a, b) = \sum_{i=1}^{n} [(ax_i + b) - y_i]^2 \tag{7.3}$$

with respect to the parameters a and b. In view of (7.2) we have

$$\begin{aligned}
\frac{\partial}{\partial a} \sum_{i=1}^{n} [(ax_i + b) - y_i]^2 &= 2 \sum_{i=1}^{n} (ax_i + b - y_i)(x_i) = 0 \\
\frac{\partial}{\partial b} \sum_{i=1}^{n} [(ax_i + b) - y_i]^2 &= 2 \sum_{i=1}^{n} (ax_i + b - y_i) = 0
\end{aligned}$$

Summing term by term leads to the so-called **normal equations**

$$\begin{aligned}
a \sum_{i=1}^{n} x_i^2 + b \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i \\
a \sum_{i=1}^{n} x_i + bn &= \sum_{i=1}^{n} y_i
\end{aligned} \tag{7.4}$$

The second term, bn, in the second equation comes from the fact that $\sum_{i=1}^{n} b = nb$. The solution of this linear system is

$$\begin{aligned}
a &= \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} \\
b &= \frac{\sum x_i^2 \sum y_i - \sum x_i y_i \sum x_i}{n \sum x_i^2 - (\sum x_i)^2}
\end{aligned} \tag{7.5}$$

where $\sum = \sum_{i=1}^{n}$.

m-function 7.1

The following MATLAB function **linlsqr.m** constructs a least squares line $y = ax + b$ that best fits a table of x and y values. INPUTS are x and y row vectors.

```
function linlsqr(x,y)
% Construct a least square line y=ax+b
% that fits x and y row or column vectors.
n=length(x);
n=length(y);
sumx=sum(x);
sumy=sum(y);
sumx2=sum(x.*x);
sumxy=sum(x.*y);
a=(n*sumxy-sumx*sumy)/(n*sumx2-sumx^2);
b=(sumx2*sumy-sumxy*sumx)/(n*sumx2-sumx^2);
disp('              linear least squares')
fprintf('\n a =%12.6f\n',a)
fprintf(' b =%12.6f\n',b)
disp('_____________________________________________')
disp('        x        y          a*x+b          |y-(ax+b)|   ')
disp('_____________________________________________')
for i=1:n
  yl=a*x(i)+b;
  err(i)=abs(yl-y(i));
  fprintf('%6.2f %6.2f  %12.6f  %12.6f\n',x(i),y(i),yl,err(i))
end
err=sum(err.*err);
fprintf('\n E(a,b) =%12.6f\n',sum(err))
```

EXAMPLE 7.1

Using the method of least squares, find the linear function that best fits the following data (see Figure 7.3).

x	1	1.5	2	2.5	3	3.5	4
y	25	31	27	28	36	35	32

Using values in Table 7.3, we get

$$a = \frac{7(554) - (17.5)(214)}{7(50.75) - (17.5)^2} = 2.714\,285\,71$$

$$b = \frac{(50.75)(214) - (17.5)(554)}{7(50.75) - (17.5)^2} = 23.78571429.$$

Therefore, the least squares line is

$$y = 2.71428571x + 23.78571429$$

```
» x=[1:0.5:4];
» y=[25 31 27 28 36 35 32];
» linlsqr(x,y)

            linear least squares

a =   2.714286
b =   23.785714
```

x	y	a*x+b	|y-(ax+b)|
1.0	25	26.500000	1.500000
1.5	31	27.857143	3.142857
2.0	27	29.214286	2.214286
2.5	28	30.571429	2.571429
3.0	36	31.928571	4.071429
3.5	35	33.285714	1.714286
4.0	32	34.642857	2.642857

E(a,b) = 50.142857

Table 7.3 Linear least squares for Example 7.1.

EXERCISE SET 7.1

1. Find the linear function that best fits the data

x	1.2	2.2	3.2	4.2	5.2
y	2.7	4.2	5.8	9.1	10.1

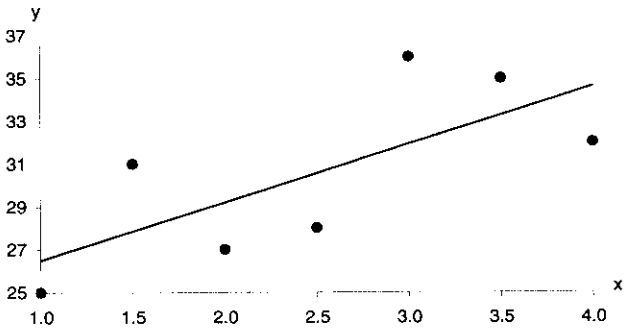


FIGURE 7.3
Linear least squares $y = 2.7143x + 23.7857$ for Example 7.1.

by using the method of least squares and compute the error.

2. Given the data,

x	1	1.5	2	2.5	3	3.5	4
y	25	31	27	28	36	35	34

fit a straight line to the data using the method of least squares and compute the error.

3. The following table lists the homework grades, out of 100 students. Find the linear function that best fits the data, using the method of least squares and compute the error.

Homework #	1	2	3	4	5	6	7	8	9	10	11	12
Grades	78	65	92	57	69	60	80	91	60	70	55	45

Use this function to predict the grade of the student on the next homework, # 13.

4. The following table lists the temperatures of a room recorded during the time interval [1:00,7:00]. Find the best linear least squares that approximate the table.

Time	1:00	2:00	3:00	4:00	5:00	6:00	7:00
Temperature	13	15	20	14	15	13	10

Use your result to predict the temperature of the room at 8:00.

5. **Computer assignment:**
Write a computer program in FORTRAN or C to construct the linear least squares that best fit a given set of data points.
Input data to the program should be

   (a) m data points (x_i, y_i).

   Output should consist of:

   (a) The coefficient a and b of the linear least squares $f(x) = ax + b$.

   (b) The error.

   Test your program to solve Exercise 1.

7.2 LEAST SQUARES POLYNOMIAL

The method of least squares data fitting is not restricted to linear functions $f(x) = ax + b$ only. As a matter of fact, in many cases data from experimental results are not linear, so we need to consider some other guess functions. To illustrate this, suppose that the guess function for the data in Table 7.2 is a polynomial

$$p_m(x) = \sum_{k=0}^{m} a_k x^k \tag{7.6}$$

of degree $m \leq n - 1$. So, according to the least squares principle, we need to find the coefficients $a_0, a_1, \ldots, a_m$ that minimize

$$\begin{aligned} E(a_0, \ldots, a_m) &= \sum_{i=1}^{n} [p_m(x_i) - y_i]^2 \\ &= \sum_{i=1}^{n} \left[\sum_{k=0}^{m} (a_k x_i^k) - y_i \right]^2 . \end{aligned} \tag{7.7}$$

As before, E is minimum if

$$\frac{\partial}{\partial a_j} E(a_0, \ldots, a_m) = 0, \quad j = 0, 1, \ldots, m \tag{7.8}$$

that is

$$\begin{aligned}
\frac{\partial E}{\partial a_0} &= \sum_{i=1}^{n} 2 \left[\sum_{k=0}^{m} (a_k x_i^k) - y_i \right] = 0 \\
\frac{\partial E}{\partial a_1} &= \sum_{i=1}^{n} 2 \left[\sum_{k=0}^{m} (a_k x_i^k) - y_i \right] (x_i) = 0 \\
\vdots &= \vdots \\
\frac{\partial E}{\partial a_m} &= \sum_{i=1}^{n} 2 \left[\sum_{k=0}^{m} (a_k x_i^k) - y_i \right] (x_i^m) = 0
\end{aligned} \tag{7.9}$$

Rearranging Eqn. (7.9) gives the $(m+1)$ normal equations for the $(m+1)$ unknowns $a_0, a_1, \ldots, a_m$

$$\begin{array}{llll}
a_0 n & + a_1 \sum x_i & + \cdots + a_m \sum x_i^m & = \sum y_i \\
a_0 \sum x_i & + a_1 \sum x_i^2 & + \cdots + a_m \sum x_i^{m+1} & = \sum y_i x_i \\
a_0 \sum x_i^2 & + a_1 \sum x_i^3 & + \cdots + a_m \sum x_i^{m+2} & = \sum y_i x_i^2 \\
\vdots & + \quad \vdots & + \cdots + \quad \vdots & = \quad \vdots \\
a_0 \sum x_i^m & + a_1 \sum x_i^{m+1} & + \cdots + a_m \sum x_i^{2m} & = \sum y_i x_i^m
\end{array} \tag{7.10}$$

As before, $\sum$ denotes $\sum_{i=1}^{n}$. It can be shown that these normal equations have a unique solution provided that the x_i's are distinct. However, when the x-values are equally spaced, the matrix of this linear system is a Hilbert matrix (see Exercise 5 Section 4.2) which is extremely ill-conditioned. The degree of the approximating polynomial is severely limited in most cases by the round-off error. For example, using a computer with precision of about eight decimal digits will usually produce meaningless results if n is greater than six.

EXAMPLE 7.2

Find the least squares polynomial of degree three that fit the following table of values.

x	0.0	0.5	1.0	1.5	2.0	2.5
y	0.0	0.20	0.27	0.30	0.32	0.33

m-function 7.2

The following MATLAB function **polylsqr.m** constructs a least squares polynomial of degree *m* that best fits a table of *x* and *y* values. INPUTS are *x* and *y* row vectors and the degree *m* of the polynomial.

```
function polylsqr(x,y,m)
% Construct the least squares polynomial of degree m
% that fits x and y row or column vectors.
n=length(x);
n=length(y);
for k=1:2*m+1
  c(k)=sum(x.^(k-1));
end
%Find the coefficient vector b of normal equations
for k=1:m+1
  b(k)=sum(y.*x.^(k-1));
end
% Find the coefficient matrix A of normal equations
for i=1:m+1
  for j=1:m+1
    A(i,j)=c(j+i-1);
  end
end
fprintf('\n')
disp(' The Augmented Matrix of the normal equations, [A b] =')
fprintf('\n')
disp([A b'])
z=A\b';
disp('The coeff.  a0,...,an of the least squares polynomial are ')
fprintf('\n')
disp(z')
% Evaluate the polynomial at xi, i=1,..,n.
disp('_____________________________________________')
disp('        xi        yi        p(xi)        |yi-p(xi)|   ')
disp('_____________________________________________')
for i=1:n
  s=z(1);
  for k=2:m+1
    s=s+z(k)*x(i)^(k-1);
  end
  p(i)=s;
  err(i)=abs(y(i)-p(i));
```

```
  fprintf('%6.2f  %6.2f  %12.6f  %12.6f\n',x(i),y(i),p(i),err(i))
end
err=sum(err.*err);
fprintf('\n E(a0,...,an) =%12.6g\n',sum(err))
[x' y' p']
```

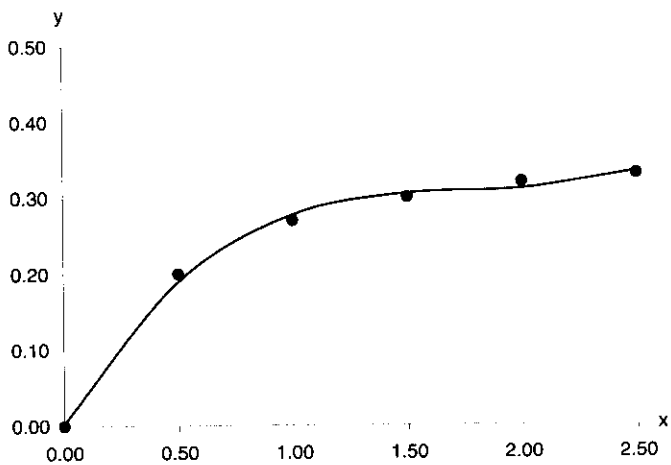


FIGURE 7.4
Polynomial least squares for Example 7.2.

From Eqn. (7.10), the normal equations are

$$\begin{bmatrix} 6.0000 & 7.5000 & 13.750 & 28.1250 \\ 7.5000 & 13.7500 & 28.1250 & 61.1875 \\ 13.7500 & 28.1250 & 61.1875 & 138.2813 \\ 28.1250 & 61.1875 & 138.2813 & 320.5469 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1.4200 \\ 2.2850 \\ 4.3375 \\ 9.0237 \end{bmatrix}$$

The solution of this linear system is

$$a_0 = 0.0033,\ a_1 = 0.4970,\ a_2 = -0.2738,\ a_3 = 0.0511$$

and the desired polynomial is

$$p_3(x) = 0.0033 + 0.4970x - 0.2738x^2 + 0.0511x^3.$$

The computed results are shown in Table 7.4.

```
» x=[0:0.5:2.5];
» y=[0 0.2 0.27 0.3 0.32 0.33];
» polylsqr(x,y,3)

The Augmented Matrix of the normal equations, [A b] =

    6.000     7.500    13.750    28.125     1.420
    7.500    13.750    28.125    61.188     2.285
   13.750    28.125    61.188   138.281     4.338
   28.125    61.188   138.281   320.547     9.024

The coefficients ao,....an of the least-squares polynomial are

  0.0033   0.4970  -0.2738   0.0511

   xi        yi        p(xi)      |yi-p(xi)|
  0.00      0.00      0.003333   0.003333
  0.50      0.20      0.189762   0.010238
  1.00      0.27      0.277619   0.007619
  1.50      0.30      0.305238   0.005238
  2.00      0.32      0.310952   0.009048
  2.50      0.33      0.333095   0.003095

          E(ao,...,an) = 0.000292857
```

Table 7.4 Polynomial least squares for Example 7.2.

In the following example we illustrate the use of the MATLAB function polylsqr to higher degree polynomial.

EXAMPLE 7.3

Find the least squares polynomial of degree four that fit the data given by the vectors

x = -3:0.4:3.2

y = sin(x)

By calling the MATLAB function polysqrt(x,y,4), we obtain the results shown in Table 7.5 and represented by Figure 7.5.

```
» x=[-3.0: 0.4: 3.3];
» y=sin(x);
» polylsqr(x,y,4)

The Augmented Matrix of the normal equations, [A b] =

1.0e+004 *
  0.0016   0.0000   0.0054  -0.0000   0.0331  -0.0000
  0.0000   0.0054  -0.0000   0.0331        0   0.0016
  0.0054  -0.0000   0.0331        0   0.2388        0
 -0.0000   0.0331        0   0.2388        0   0.0062
  0.0331        0   0.2388   0.0000   1.8648  -0.0000

The coefficients a0,...,an of the least squares polynomial are

 -0.0000   0.8552   0.0000  -0.0928  -0.0000

 xi      yi      p(xi)       |yi-p(xi)|

-3.00   -0.14   -0.060136   0.080984
-2.60   -0.52   -0.592535   0.077033
-2.20   -0.81   -0.893326   0.084830
-1.80   -0.97   -0.998142   0.024295
-1.40   -0.99   -0.942615   0.042835
-1.00   -0.84   -0.762376   0.079095
-0.60   -0.56   -0.493058   0.071585
-0.20   -0.20   -0.170291   0.028378
 0.20    0.20    0.170291   0.028378
 0.60    0.56    0.493058   0.071585
 1.00    0.84    0.762376   0.079095
 1.40    0.99    0.942615   0.042835
 1.80    0.97    0.998142   0.024295
 2.20    0.81    0.893326   0.084830
 2.60    0.52    0.592535   0.077033
 3.00    0.14    0.060136   0.080984

E(a0,...,an) =  0.0685988
```

Table 7.5 Polynomial least squares for Example 7.3.

The principal of least squares, applied to polynomials given by Eqn. (7.6), can be extended to a much more general setting. Let

$$p(x) = \sum_{k=1}^{m} c_k \psi_k(x)$$

where the ψ's are polynomials of degree $\leq k$.

The polynomials ψ_k, called "basis functions," are known and held fixed. So, according to the least squares principle, we need to find

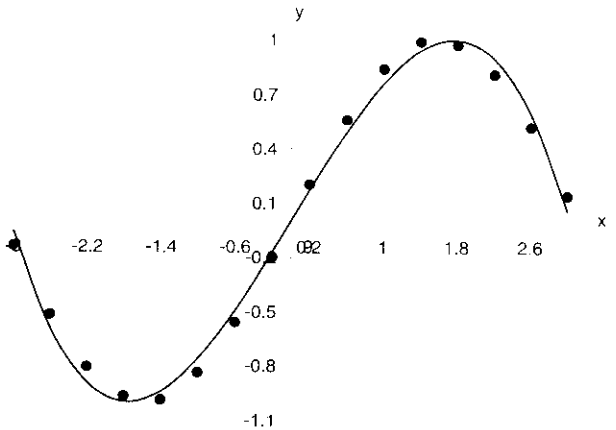


FIGURE 7.5
Polynomial of degree 4 that fits the data for Example 7.3.

$c_1, c_2, \ldots, c_m$ that minimize

$$E(c_1, \ldots, c_m) = \sum_{k=1}^{n} \left[\sum_{i=1}^{m} c_i \psi_i(x_k) - y_k \right]^2 .$$

Proceeding as before, E is minimum if

$$\frac{\partial E}{\partial c_j} = \sum_{k=1}^{n} 2 \left[\sum_{i=1}^{m} c_i \psi_i(x_k) - y_k \right] \psi_j(x_k) = 0, \quad j = 1, \ldots, m$$

Thus,

$$\sum_{i=1}^{m} \left[\sum_{k=1}^{n} \psi_i(x_k) \psi_j(x_k) \right] c_i = \sum_{k=1}^{n} y_k \psi_j(x_k) \quad j = 1, \ldots, m.$$

Hence $p(x)$ is found by solving these m linear normal equations for the m unknowns $c_1, c_2, \ldots, c_m$.

If the functions $\psi_1, \psi_2, \ldots, \psi_m$ have the property of *orthogonality*, that is

$$\sum_{k=1}^{n} \psi_i(x_k) \psi_j(x_k) = 0 \quad \text{if} \quad i \neq j$$

then the normal equations reduce to

$$\sum_{k=1}^{n} [\psi_j(x_k)]^2 c_j = \sum_{k=1}^{n} y_k \psi_j(x_k), \quad j = 1, \ldots, m.$$

Thus, the least squares approximation problem is greatly simplified in this case.

An example of orthogonal polynomials is given by the *Legendre polynomials* defined on the interval $[-1, 1]$. The first few Legendre polynomials are

$$\begin{aligned} L_1(x) &= 1, \\ L_2(x) &= x, \\ L_3(x) &= \frac{1}{2}(3x^2 - 1), \\ L_4(x) &= \frac{1}{2}(5x^3 - 3x), \\ L_5(x) &= \frac{1}{8}(35x^4 - 30x^2 + 3). \end{aligned}$$

These polynomials can be generated by the recursion relation

$$L_n(x) = \frac{2n-3}{n-1} x L_{n-1}(x) - \frac{n-2}{n-1} L_{n-2}(x), \quad n \geq 3.$$

The use of these polynomials is not restricted to the interval $[-1, 1]$, since any finite interval [a,b] can be transformed into $[-1, 1]$ by the change of variables

$$z_k = \frac{1}{b-a}(2x_k - a - b), \quad k = 1, \ldots, n.$$

Experience has shown that, contrarily to the polynomials in (7.6), the Legendre polynomials are a satisfactory choice for the ψ's and the normal equations are reasonably well-conditioned. More examples and details on orthogonal polynomials can be found in [5].

EXERCISE SET 7.2

1. Use the data of Exercise 1 in Section 7.1 to compute the least squares polynomial of degree two that best fits the data.

2. Given the data,

x	1.0	1.2	1.5	1.8	2.1	2.4	2.8
y	5.5	7.3	14.1	26.3	48.2	80.1	111.6

(a) Find the least squares polynomial of degree two that best fits the data and compute the error.

(b) Find the least squares polynomial of degree three that best fits the data and compute the error.

3. Given the following data

x	1.2	3.0	4.5	5.6	6.9	8.1
y	2.3	3.3	5.6	15.4	38.2	59.1

find the least squares polynomial of degree two that best fits the data. Use the resulting polynomial to approximate $f(6.0)$.

4. Suppose the quadratic function

$$f(x) = x^2 + 3x + 7$$

is used to generate the data,

x	0	1	2	3
$f(x)$	7	11	17	25

find the least squares polynomials of degree two and three that best fit these data. Compute these polynomials at the four values of x and compare your results with the exact ones.

7.3 NONLINEAR LEAST SQUARES

The actual selection of the **guess function** to be used for the least squares fit depends on the nature of the experimental data. So far, we have studied only least squares polynomials. However, many cases arise when data from experimental tests is exponentially related or tends to have vertical and horizontal asymptotes, so we need to consider guess functions other than polynomials. In this section, we broaden our discussion of least squares approximations to some popular forms of guess functions that arise when data are plotted.

7.3.1 Exponential form

Suppose we want to fit the data in Table 7.2 by a function of the form

$$f(x) = ae^{bx} \tag{7.11}$$

in the least squares sense. If we develop the normal equations for (7.11) using (7.8), we will end up with nonlinear simultaneous equations which are more difficult to solve than linear equations. So, one way to get around this difficulty is to first linearize the exponential form (7.11) by taking the logarithm of $f(x)$:

$$\ln f(x) = bx + \ln a$$

we now set

$$F(x) = \ln f(x), \quad \alpha = \ln a \quad \text{and} \quad \beta = b$$

to get a linear function of x

$$F(x) = \beta x + \alpha. \tag{7.12}$$

Finally, transform the points (x_i, y_i) in Table 7.2 to the points $(x_i, \ln y_i)$ and use the linear least squares described in Section 7.1 to get β and α. Having obtained β and α, we use the relations

$$b = \beta \quad \text{and} \quad a = e^{\alpha}$$

to obtain a and b.

EXAMPLE 7.4

Find the least squares exponential that best fits the following data:

x	1	3	4	6	9	15
y	4.0	3.5	2.9	2.5	2.75	2.0

Apply the transformation (7.12) to the original points and obtain

$X = x$	1.0	3.0	4.0	6.0	9.0	15.0
$Y = \ln y$	1.39	1.25	1.06	0.92	1.01	0.69

m-function 7.3

The following MATLAB function **explsqr.m** constructs a least squares exponential $y = ae^{bx}$ that best fits a table of *x* and *y* values. INPUTS are *x* and *y* row vectors.

```
function explsqr(x,y)
% Construct the least square exponential y=a*exp(b*x)
% that fits x and y row or column vectors.
n=length(x);
n=length(y);
z=log(y);
sumx=sum(x);
sumz=sum(z);
sumx2=sum(x.*x);
sumxz=sum(x.*z);
beta=(n*sumxz-sumx*sumz)/(n*sumx2-sumx^2);
b=beta;
alfa=(sumx2*sumz-sumxz*sumx)/(n*sumx2-sumx^2);
a=exp(alfa);
disp('              Exponential least squares')
fprintf('\n a =%12.6f\n',a)
fprintf(' b =%12.6f\n',b)
disp('_____________________________________________________')
disp('    xi      yi      ln(yi)     a*exp(b*xi)      |yi-[a*exp(b*xi)]|    ')
disp('_____________________________________________________')
for i=1:n
  ye=a*exp(b*x(i));
  err(i)=abs(ye-y(i));
  fprintf('%6.2f  %6.2f  %6.2f  %12.6f
%12.6f\n',x(i),y(i),z(i),ye,err(i))
end
```

The transformed points are shown in Table 7.6. The equation of the least squares exponential is

$$y = 3.801404\ e^{-0.044406x}$$

```
» x=[1 3 4 6 9 15];
» y=[4 3.5 2.9 2.5 2.75 2];
» explsqr(x,y)

          Exponential least squares

a =   3.801404
b =  -0.044406
```

xi	yi	ln(yi)	a*exp(b*xi)	|yi-[a*exp(b*xi)]|
1	4.00	1.39	3.636293	0.363707
3	3.50	1.25	3.327274	0.172726
4	2.90	1.06	3.182757	0.282757
6	2.50	0.92	2.912280	0.412280
9	2.75	1.01	2.549046	0.200954
15	2	0.69	1.952841	0.047159

Table 7.6 Exponential least squares for Example 7.4.

7.3.2 Hyperbolic form

Suppose we want to fit the data in Table 7.2 by a function of the form

$$f(x) = a + \frac{b}{x} \tag{7.13}$$

in the least squares sense. This guess function is useful when the data tend to have vertical and horizontal asymptotes. Again, before finding the normal equations, we linearize $f(x)$ by setting

$$F(x) = f(x), \quad \alpha = a, \quad \beta = b, \quad \text{and} \quad X = \frac{1}{x}.$$

Thus,

$$F(X) = \alpha + \beta X. \tag{7.14}$$

Again (7.14) is now a linear function in X, and if we transform the points (x_i, y_i) in Table 7.2 to $(1/x_i, y_i)$, we can use the linear least squares to find $\alpha = a$ and $\beta = b$.

EXAMPLE 7.5

Find the least squares hyperbolic that best fits the data in Example 7.4.

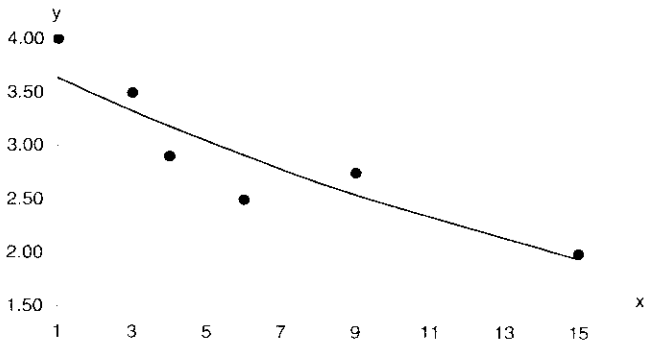


FIGURE 7.6
Exponential least squares for Example 7.4.

Apply the transformation (7.14) to the original points and obtain

$X = 1/x$	1	0.33	0.25	0.17	0.11	0.07
Y	4	3.5	2.9	2.5	2.75	2.0

The transformed points are shown in Table 7.7. The equation of the least squares hyperbolic is

$$y = 2.364128 + \frac{1.797526}{x}.$$

We close this section by pointing out that the approximation obtained in this manner for both of these examples is not the least squares approximation for the original problem, and it can, in some cases, differ from the least squares approximation to the original problem.

EXERCISE SET 7.3

1. Find the least squares exponential that best fits the following data

x	0.0	1.5	2.5	3.5	4.5
y	2.0	3.6	5.4	8.1	12.0

m-function 7.4

The following MATLAB function **hyplsqr.m** constructs a least squares hyperbolic $y = a + b/x$ that best fits a table of x and y values. INPUTS are x and y row vectors.

```
function hyplsqr(x,y)
% Construct the least square hyperbolic y=a+b/x
% that fits x and y row or column vectors.
n=length(x);
n=length(y);
z=ones(size(x));
z=z./x;
sumz=sum(z);
sumy=sum(y);
sumz2=sum(z.*z);
sumzy=sum(z.*y);
b=(n*sumzy-sumz*sumy)/(n*sumz2-sumz^2);
a=(sumz2*sumy-sumzy*sumz)/(n*sumz2-sumz^2);
disp('              hyperbolic least squares')
fprintf('\n a =%12.6f\n',a)
fprintf(' b =%12.6f\n',b)
disp('_____________________________________________')
disp('      x      y      1/xi      a+b/x      |y-(a+b/xi)|  ')
disp('_____________________________________________')
for i=1:n
  yh=a+b/x(i);
  err(i)=abs(yh-y(i));
  fprintf('%6.2f %6.2f %6.2f %12.6f  %12.6f\n',x(i),y(i),z(i),yh,err(i))
end
```

Use your result to predict the value of y when $x = 5.0$.

2. Find the least squares hyperbolic that best fits the data

x	1.0	1.4	1.8	2.2	2.6
y	3.7	3.5	3.4	3.1	2.9

Use your result to predict the value of y when $x = 3.0$.

3. Given a table of data points (x_i, y_i) for $1 \leq i \leq n$, find the normal equations for the following guess functions

   (a) $f(x) = a + b \ln x$

```
x=[1 3 4 6 9 15];
» y=[4 3.5 2.9 2.5 2.75 2];
» hyplsqr(x,y)

            Hyperbolic least squares

a =   2.364128
b =   1.797526
```

xi	yi	1/xi	a+b/xi	|yi-(a+b/xi)|
1	4.00	1.000	4.161654	0.161654
3	3.50	0.330	2.963304	0.536696
4	2.90	0.250	2.81351	0.08649
6	2.50	0.170	2.663716	0.163716
9	2.75	0.110	2.563853	0.186147
15	2.00	0.070	2.483963	0.483963

Table 7.7 Hyperbolic least squares for Example 7.5.

(b) $f(x) = \frac{1}{a+x}$

(c) $f(x) = ax^b$

4. Find the normal equations for fitting a curve of the form

$$y = a \sin bx$$

5. Find the normal equations for fitting a curve of the form

$$f(x) = ax^2 + bx + c$$

6. Find the best function of the form $f(x) = a + b \sin \frac{\pi x}{10}$ that best fits the data

x	0	1.0	2.3	4.0	5.1	6.0	6.5	7.0	8.1	9.0
$f(x)$	0.2	0.8	2.5	4.3	3.0	5.0	3.5	2.4	1.3	2.0
x	9.3	11.0	12.1	13.1	14.0	16	17.5	17.8	19.0	20.0
$f(x)$	-0.3	-1.3	-4.0	-4.9	-4.0	-3.0	-3.5	-1.6	-1.4	-0.1

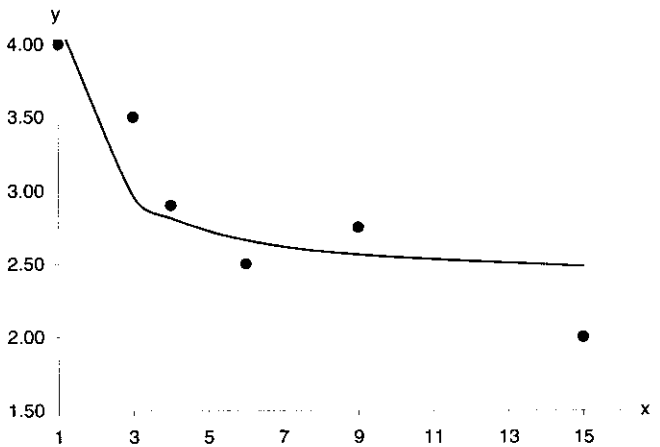


FIGURE 7.7
Hyperbolic least squares for Example 7.5.

7.4 TRIGONOMETRIC LEAST SQUARES POLYNOMIAL

In this section we shall discuss how the least squares principle may be applied to trigonometric curves of the form

$$p(x) = \frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x) + \sum_{k=1}^{m} b_k \sin(k\omega x) \tag{7.15}$$

where ω is given. (7.15) is called a **trigonometric polynomial of order m**.

Suppose we want to approximate the data in Table 7.2 with the function (7.15) using the least squares method. The procedure is handled in a similar manner and requires choosing the constants $a_0, \ldots, a_m, b_1, \ldots, b_m$ that minimize the least squares error

$$\begin{aligned} E &= \sum_{i=1}^{n} [p(x_i) - y_i]^2 \\ &= \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n} p(x_i)y_i + \sum_{i=1}^{n} [p(x_i)]^2 \end{aligned}$$

$$= \sum_{i=1}^{n} y_i^2 - 2\sum_{i=1}^{n}\left[\frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x_i) + \sum_{k=1}^{m} b_k \sin(k\omega x_i)\right] y_i$$
$$+ \sum_{i=1}^{n}\left[\frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x_i) + \sum_{k=1}^{m} b_k \sin(k\omega x_i)\right]^2$$

where $E = E(a_0, \ldots, a_m, b_1, \ldots, b_m)$.

Proceeding as before, for E to be minimized, it is necessary that

$$\frac{\partial E}{\partial a_j} = 0 \quad \text{for} \quad j = 0, \ldots, m$$

and

$$\frac{\partial E}{\partial b_j} = 0 \quad \text{for} \quad j = 1, \ldots, m$$

Taking the partial derivatives and rearranging terms, we get

$$\sum_{i=1}^{n}\left[\frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x_i) + \sum_{k=1}^{m} b_k \sin(k\omega x_i)\right] = \sum_{i=1}^{n} y_i$$
$$\sum_{i=1}^{n}\left[\frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x_i) + \sum_{k=1}^{m} b_k \sin(k\omega x_i)\right] \cos(j\omega x_i) = \sum_{i=1}^{n} \cos(j\omega x_i) y_i \tag{7.16}$$
$$\sum_{i=1}^{n}\left[\frac{a_0}{2} + \sum_{k=1}^{m} a_k \cos(k\omega x_i) + \sum_{k=1}^{m} b_k \sin(k\omega x_i)\right] \sin(j\omega x_i) = \sum_{i=1}^{n} \sin(j\omega x_i) y_i$$

for $j = 1, \ldots, m$.

These are the normal equations which form a linear system of $(2m+1)$ equations in $(2m+1)$ unknowns and which can be solved using Gaussian elimination. The derivation of the coefficients a_i and b_i is usually called *discrete Fourier analysis.*

EXAMPLE 7.6

Find the trigonometric least squares polynomial that approximate the following data:

x	0.0	0.4	0.8	1.2	1.6	2.0	2.4	2.8	3.2	3.6	4.0	4.4
y	3.0	2.1	1.3	0.5	0	-0.2	0	0.3	1.1	2.0	2.9	3.5

using $m = 1$ and $\omega = 1$.

From Eqn. (7.16), we get the following normal equations

$$\begin{aligned} 6a_0 + \left[\sum \cos x_i\right] a_1 + \left[\sum \sin x_i\right] b_1 &= \sum y_i \\ \left[\sum \cos x_i\right] a_0 + \left[\sum \cos^2 x_i\right] a_1 + \left[\sum \cos x_i \sin x_i\right] b_1 &= \sum y_i \cos x_i \\ \left[\sum \sin x_i\right] a_0 + \left[\sum \cos x_i \sin x_i\right] a_1 + \left[\sum \sin^2 x_i\right] b_1 &= \sum y_i \sin x_i \end{aligned}$$

with $\sum = \sum_{i=1}^{12}$.

By substituting the quantities

$$\begin{aligned} \sum \cos x_i &= -2.000867, & \sum \sin x_i &= 2.748837, \\ \sum \sin x_i \cos x_i &= 1.217139, & \sum y_i &= 16.5, \\ \sum \cos^2 x_i &= 6.393092, & \sum \sin^2 x_i &= 5.606908, \\ \sum y_i \sin x_i &= -4.339589, & \sum y_i \cos x_i &= -0.041184 \end{aligned}$$

into the normal equations, we get the system

$$\begin{aligned} 6.000000a_0 + -2.000867a_1 + 2.748837b_1 &= 16.5 \\ -2.000867a_0 + 6.393092a_1 + 1.217139b_1 &= -0.041184 \\ 2.748837a_0 + 1.217139a_1 + 5.606908b_1 &= -4.339589 \end{aligned}$$

The solution of this system is

$$a_0 = 5.352575, \quad a_1 = 2.415546, \quad b_1 = -3.922482$$

Thus,

$$p(x) = 2.6762875 + 2.415546 \cos x - 3.922482 \sin x.$$

EXERCISE SET 7.4

1. Derive the normal equations for fitting the trigonometric curve

$$p(x) = a_0 + a_1 \cos x + b_1 \sin x$$

Find the coefficients a_0, a_1, and b_1 that best fit the data

x	0	1	2	3
$f(x)$	5	54	99	181

in the least squares sense.

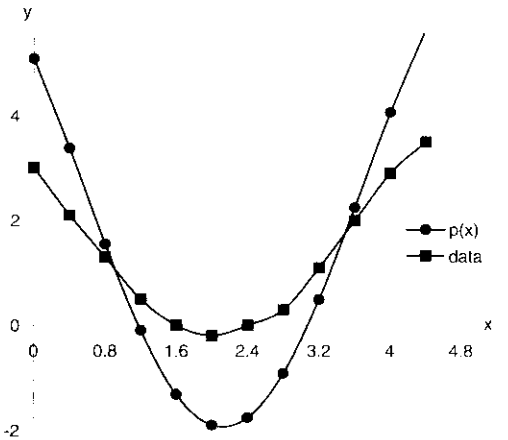


FIGURE 7.8
Plot of $f(x)$ and $p(x)$ in $[0, 3]$ for Example 7.6.

APPLIED PROBLEMS FOR CHAPTER 7

1. Table 7.8 lists the Gross National Product (GNP) in current dollars and constant dollars. Current dollars are simply the dollar value with no adjustment for inflation. Constant dollars represent the GNP based on the value of the dollar in 1958.

   (a) Determine graphically which guess function seems best-suited to fit these data and find the guess function using the method of least squares.

   (b) Use the results of part (a) to predict the two GNPs in 1990 and 2000.

2. The following table lists the total water usage in the United States in billions of gallons per day.

Year	1930	1940	1950	1960	1970
Water Use	110.2	137.4	202.6	322.7	411.1

   (a) Find the least squares exponential of the water consumption on time.

Year	GNP(Current Dollars, $ millions)	GNP(Constant Dollars, $ millions)
1950	284.7	355.2
1955	398.1	438.1
1960	503.6	487.6
1965	684.8	617.7
1966	749.8	658.0
1967	763.4	674.5
1968	865.6	707.5

Table 7.8 GNP from 1950 t0 1968.

(b) Use the results of part (a) to predict the water consumption in 1980 and 1990.

3. The following table lists the number of motor vehicle accidents in the United States for various years from 1950 to 1968.

Year	1950	1955	1960	1965	1966	1967	1968
No. of Accidents (in thousands)	8,200	9,800	10,300	13,100	13,500	13,600	14,500
Accidents per 10,000 Vehicles	1,687	1,576	1,396	1,438	1,418	1,384	1,414

(a) Find the linear least squares of the number of accidents on time. Use it to predict the number of accidents in 1990.

(b) Compute the quadratic least squares of the number of accidents per 10,000 vehicles on time. Use it to predict the number of accidents per 10,000 vehicles in 1990.

4. A biologist is doing an experiment on the growth of a certain bacteria culture. After 4 hours the following data has been recorded:

t	0	1	2	3	4
p	1.0	1.8	3.3	6.0	11.0

where t is the number of hours and p the population in thousands. Determine the least squares exponential that best fits these data. Use your results to predict the population of bacteria after 5 hours.

Chapter 8

Numerical Differentiation and Integration

This chapter is split into two parts, the first dealing with the problem of estimating the derivative of a function, and the second with numerical methods for approximating the definite integral

$$I = \int_a^b f(x)dx \tag{8.1}$$

with $[a, b]$ finite. Several formulas for approximating a first and second derivative by a difference quotient are given. Next, we will look at several methods of evaluating definite integrals numerically.

In most cases, the evaluation of definite integrals are either impossible, or else very difficult, to evaluate analytically. It is known from calculus that the integral I represents the area of the region between the curve $y = f(x)$ and the lines $x = a$ and $x = b$. So the basic idea of approximating I is to replace $f(x)$ by an interpolating polynomial $p(x)$ and then integrate the polynomial by finding the area of the region bounded by $p(x)$, the lines $x = a$ and $x = b$ and the x-axis. This process is called **numerical quadrature**. The integration rules that we will be developing correspond to the varying degree of the interpolating polynomials. The most important are the ones with degree one, two, and three. However, care must be exercised when using interpolating polynomials of higher degree; round-off errors and irregularities can cause a problem in these cases.

8.1 NUMERICAL DIFFERENTIATION

In many practical cases, we are faced with the problem of finding the derivative of a function whose form is either known only as a tabulation of data, or not practical to use for calculations. In this section, we will study some techniques for the numerical computation of first and second derivatives. These results will also be used to obtain the numerical solution of differential equations that will be presented in Chapters 9 and 12.

We begin with the series expansion of $f(x)$ about x. We assume that $f(x)$ has as many continuous derivatives as may be required. From Taylor's formula

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \cdots. \tag{8.2}$$

Solving Eqn. (8.2) for $f'(x)$ yields

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \cdots \tag{8.3}$$

so that an approximation for $f'(x)$ may be written as

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}, \quad \text{for a small value of } h. \tag{8.4}$$

The expression for $f'(x)$ in Eqn. (8.4) is called the **forward divided difference** approximation. Graphically, it represents the slope of the line passing through the points $(x, f(x))$ and $(x+h, f(x+h))$. Another approximation for $f'(x)$ can be obtained by replacing h by $-h$ in Eqn. (8.2). That is,

$$f'(x) \approx \frac{f(x) - f(x-h)}{h}, \quad \text{for a small value of } h. \tag{8.5}$$

Eqn. (8.5) is called the **backward divided difference** approximation to $f'(x)$.

Note that, from Eqn. (8.3), the **truncation error** for both the forward and backward difference approximation to $f'(x)$ is

$$-\frac{h}{2}f''(\zeta)$$

for some ζ in the interval $(x, x+h)$, for the forward difference and for some ζ in $(x-h, x)$, for the backward difference.

One can see that for a linear function, the approximations are exact since the error term is zero in this case.

Again, if we replace h by $-h$ in Eqn. (8.2), and then subtract the resulting equation from the old one, we obtain a very popular formula for approximating $f'(x)$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{3!}f'''(x) - \frac{h^4}{5!}f^{(5)} - \cdots. \tag{8.6}$$

That is,

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad \text{for a small value of } h. \tag{8.7}$$

Formula (8.7) is known as the **central difference** approximation of $f'(x)$. Hence, we have an approximation to $f'(x)$ with an error of the order h^2.

EXAMPLE 8.1

Given $f(x) = e^x$, approximate $f'(1.5)$ using formulas (8.7) and (8.4) with $h = 0.1$. Compare the results with the exact value $f'(x) = e^{1.5}$.

Set $x = 1.5$ and $h = 0.1$ in (8.7) and (8.4) to get

$$f'(x) \approx \frac{e^{1.6} - e^{1.4}}{0.2} \approx 4.489162287752$$

$$f'(x) \approx \frac{e^{1.6} - e^{1.5}}{0.1} \approx 4.713433540571.$$

The absolute errors are

$$|e^{1.5} - 4.489162287752| = 0.007473$$
$$|e^{1.5} - 4.713433540571| = 0.231744$$

Observe that the central difference formula gives a better approximation than the forward difference formula.

We will now proceed to find approximations for the second derivative

of $f(x)$. Again, consider the Taylor series expansion of $f(x)$ about x

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f^{(3)}(x) + \\ + \frac{h^4}{4!}f^{(4)}(x) + \cdots \tag{8.8}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f^{(3)}(x) + \\ + \frac{h^4}{4!}f^{(4)}(x) - \cdots \tag{8.9}$$

Add the two equations to get

$$f(x+h) + f(x-h) = 2f(x) + \frac{2h^2}{2!}f''(x) + \frac{2h^4}{4!}f^{(4)}(x) + \cdots.$$

We now solve for $f''(x)$ to obtain

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} - \frac{1}{12}h^2 f^{(4)}(x) + \cdots. \tag{8.10}$$

Therefore, an approximation formula for $f''(x)$ is given by

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}, \quad \text{for a small value of } h. \tag{8.11}$$

From (8.10) one can see that the truncation error is $-\frac{1}{12}h^2 f^{(4)}(\zeta)$ for some ζ in the interval $(x-h, x+h)$. Formula 8.11 is known as the **central difference formula** for the second derivative.

EXAMPLE 8.2

Let $f(x) = \sin x$. Use formulas (8.4) and (8.11) with $h = 0.1$ to approximate $f'(0.5)$ and $f''(0.5)$. Compare with the true values, $f'(0.5) = 0.87758256$ and $f''(0.5) = -0.47942554$.

Using formula (8.4) we get

$$\begin{aligned} f'(0.5) &\approx \frac{f(0.6) - f(0.5)}{0.1} \\ &\approx \frac{0.56464247 - 0.47942554}{0.1} \approx 0.8521693 \\ \text{Error} &= 0.02541326 \end{aligned}$$

Similarly, using formula (8.11), we get

$$f''(0.5) \approx \frac{f(0.6) - 2f(0.5) + f(0.4}{(0.1)^2}$$
$$\approx \frac{0.56464247 - 0.95885108 + 0.38941834}{(0.1)^2} \approx -0.479027$$

Error = 0.000399.

EXERCISE SET 8.1

1. Suppose a polynomial p_2 interpolates f at x_i, $i = 0, 1, 2$; where $x_i = a + ih$. By differentiating p_2, show that

$$p_2''(x) = \frac{f(x_1 + h) - f(x_1 - h)}{h^2}$$

2. Using Taylor's series, derive the following approximation for the third derivative of f

$$f'''(x) \approx \frac{1}{2h^3}\left[f(x + 2h) - 2f(x + h) + 2f(x - h) - f(x - 2h)\right]$$

3. Derive the following approximate formulas:

   (a) $f'(x) \approx \frac{1}{4h}\left[f(x + 2h) - f(x - 2h)\right]$

   (b) $f'(x) \approx \frac{1}{2h}\left[4f(x + h) - 3f(x) - f(x + 2h)\right]$

4. Given $f(x) = e^x$, approximate $f'(1)$ using the central difference formula with $h = 0.1$.

5. Using Taylor's series, determine the error term for the approximate formula

$$f'(x) \approx \frac{1}{2h}\left[f(x + 2h) - f(x)\right]$$

6. Repeat the previous exercise for the formula

$$f'(x) \approx \frac{1}{5h}\left[f(x + 4h) - f(x - h)\right]$$

7. Use Taylor's series to derive the following approximation formula for the third derivative of f.

$$f'''(x) \approx \frac{1}{h^3}\left[-f(x) + 3f(x+h) - 3f(x+2h) + f(x+3h)\right]$$

8. Use Taylor's series to derive the following approximation formula for the first derivative of f.

$$f'(x) \approx \frac{2f(x+3h) - 9f(x+2h) + 18f(x+h) - 11f(x)}{6h}$$

9. Show that the approximation formula in Exercise 8 has an error of $O(h^4)$.

8.2 RICHARDSON'S FORMULA

In this section, we shall develop a technique for approximating derivatives of a function f that will enable us to reduce the truncation error. This technique is known as the **Richardson extrapolation**, and has special utilities in computer programs for differentiation and integration of arbitrary functions.

To illustrate the method, let us consider Eqn. (8.6) in the form

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + c_2h^2 + c_4h^4 + \cdots \tag{8.12}$$

where $c_2, c_4, \ldots$ depend on f and x.

For x fixed, we define the function

$$a(h) = \frac{f(x+h) - f(x-h)}{2h}. \tag{8.13}$$

Now, we rewrite (8.12) as

$$f'(x) = a(h) + c_2h^2 + c_4h^4 + \cdots. \tag{8.14}$$

The idea for improving the truncation error is to try to eliminate the term c_2h^2 in Eqn. (8.14). To do so, we replace h by $h/2$ in (8.14) to get

$$f'(x) = a\left(\frac{h}{2}\right) + c_2\left(\frac{h}{2}\right)^2 + c_4\left(\frac{h}{2}\right)^4 + \cdots. \tag{8.15}$$

We can now eliminate the terms in h^2 by multiplying Eqn. (8.15) by four and subtracting Eqn. (8.14) from it. The result is

$$f'(x) = a\left(\frac{h}{2}\right) + \frac{1}{3}\left[a\left(\frac{h}{2}\right) - a(h)\right] - \frac{1}{4}c_4h^4 + \cdots. \tag{8.16}$$

One can see that we have improved the accuracy of our estimate of the derivative by reducing the error to an order h^4. The new approximation for $f'(x)$ is given by

$$f'(x) \approx a\left(\frac{h}{2}\right) + \frac{1}{3}\left[a\left(\frac{h}{2}\right) - a(h)\right]. \tag{8.17}$$

This extrapolation method can be extended since an estimate of order h^4 can be shown to have an error of order h^6 and so on.

EXAMPLE 8.3

Given $f(x) = e^x$, approximate $f'(1.5)$ using formula (8.17) with $h = 0.1$. Compare with the true value $e^{1.5}$.

Using formula (8.13) with $x = 1.5$ and $h = 0.1$, we get

$$a(h) = a(0.1) = \frac{f(1.6) - f(1.4)}{0.2} \approx 4.4889162287752$$
$$a(h/2) = a(0.05) = \frac{f(1.55) - f(1.45)}{0.1} \approx 4.483556674219.$$

In view of (8.17), we obtain

$$\begin{aligned} f'(1.5) &\approx 4.483556674219 + \frac{1}{3}(4.483556674219 - 4.4889162287752) \\ &\approx 4.481688136375. \end{aligned}$$

The error is 9.3×10^{-7}. As we can see, formula (8.17) gives a better approximation to $f'(1.5)$ than formula (8.7) for $h = 0.1$.

In general, given an approximation $a(h)$ and having computed the values

$$D_{n,1} = a\left(\frac{h}{2^{n-1}}\right), \qquad n = 1, 2, \ldots$$

for an appropriate $h > 0$, the process can be extended to m columns. Such columns and approximations are generated recursively by the formula:

$$D_{n,m+1} = \frac{4^m}{4^m - 1}D_{n,m} - \frac{1}{4^m - 1}D_{n-1,m}. \tag{8.18}$$

The truncation error associated with the entry $D_{n,m+1}$ is of order $O(h^{2m+2})$.

The procedure is best illustrated by arranging the quantities in a table of the form shown in Table 8.1.

$D_{1,1}$				
$D_{2,1}$	$D_{2,2}$			
$D_{3,1}$	$D_{3,2}$	$D_{3,3}$		
$\vdots$	$\vdots$	$\vdots$	$\ddots$	
$D_{N,1}$	$D_{N,2}$	$D_{N,3}$	$\dots$	$D_{N,N}$

Table 8.1 Two-dimensional triangular array $D_{N,N}$.

Observe that, since round-off error affects calculation by computers, computing $D_{N,N}$ for large values of N does not result, in general, to better accuracy of $D_{1,1}$. Therefore, one should put a limit on the number of repetitions of the process. The answer to the question, when one should stop the process in order to get the best approximation, is not known. On one hand, we want h small for accuracy, but on the other hand, we want h large for stability. One can empirically check whether the truncation error formula is being maintained from one level to the next. When this fails, the extrapolations should be broken off.

EXAMPLE 8.4

Given $f(x) = e^x$ and $h = 0.25$, compute $D_{6,6}$ to approximate $f'(1)$.

We have

$$D_{1,1} = a(h) = \frac{e^{1.25} - e^{0.75}}{0.5} \approx 2.7466858816$$

$$D_{2,1} = a(h/2) = \frac{e^{1.125} - e^{0.875}}{0.25} \approx 2.7253662198$$

Using formula (8.18), we get

$$D_{2,2} = \frac{4}{4-1} D_{2,1} - \frac{1}{4-1} D_{1,1} \approx 2.7182596658$$

m-function 8.1

The following MATLAB function **derive.m** Approximate the derivative of a function at a given point using formula 8.18 and the central difference approximation. INPUTS are a function *f*; a value of *h*; a specific point *a*; the number of rows *n*. The input function *f(x)* should be defined as an M-file.

```
function derive(f,h,a,n)
% Approximate the derivative of a function at x = a.
disp('     Derivative table')
disp('_____________________________________________________')
disp('  i        h        Di,1        Di,2        Di,3     ... ')
disp('_____________________________________________________')
D(1,1)=(feval(f,a+h)-feval(f,a-h))/(2*h);
fprintf('%2.0f %8.4f %12.4f\n',1,h,D(1,1));
for i=1:n-1
  h=h/2;
  D(i+1,1)=(feval(f,a+h)-feval(f,a-h))/(2*h);
  fprintf('%2.0f %8.4f %12.4f',i+1,h,D(i+1,1));
  for k=1:i
    D(i+1,k+1)=D(i+1,k)+(D(i+1,k)-D(i,k))/((4^k)-1);
    fprintf('%12.4f',D(i+1,k+1));
  end
  fprintf('\n');
end
```

Continuing in this manner leads to the values in Table 8.2.
The MATLAB function f1 used in this table is defined as follows:

```
function f=f1(x)
 f=exp(x);
```

EXERCISE SET 8.2

1. Use the MATLAB function derive to approximate the derivative of the following functions:

   (a) $f(x) = \sin x + x$ with $h = 0.3$, $N = 4$, at $x = \pi/4$.

   (b) $f(x) = xe^x - x^2$ with $h = 0.2$, $N = 4$, at $x = 2.3$.

```
» derive('f2',0.25,1,6)
  Derivative table
```

i	h	Di,1	Di,2	Di,3	...		
1	0.250000	2.746686					
2	0.125000	2.725366	2.718260				
3	0.062500	2.720052	2.718280	2.718282			
4	0.031250	2.718724	2.718282	2.718282	2.718282		
5	0.015625	2.718392	2.718282	2.718282	2.718282	2.718282	
6	0.007813	2.718309	2.718282	2.718282	2.718282	2.718282	2.718282

Table 8.2 Approximation of the derivative of $f(x) = e^x$ at $x = 1$ using Richardson's formula.

(c) $f(x) = x\cos x + 3x$ with $h = 0.2$, $N = 4$, at $x = 1.4$.

(d) $f(x) = 3x^3 - 5x^2$ with $h = 0.2$, $N = 4$, at $x = 2.6$.

(e) $f(x) = \sqrt{x^2 + 1}$ with $h = 0.5$, $N = 4$, at $x = 1.3$.

2. Given the difference quotient known as the central difference formula

$$D_h f = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$$

show that

$$\frac{4D_h f - D_{2h} f}{3}$$

is a fourth-order approximation to $f'(x)$ using Taylor's series. Assume that f is six times continuously differentiable.

3. Prove that the approximation formula

$$f'(x) \approx \frac{4f(x+h) - 3f(x) - f(x+2h)}{2h}$$

has an error that can be written as

$$f'(x) - \frac{4f(x+h) - 3f(x) - f(x+2h)}{2h} = c_1 h^2 + c_2 h^3 + \cdots$$

Determine c_1 and c_2.

4. Let $f(x) = \ln x$. Approximate $f'(1)$ for $h = 0.1$ and $h = 0.01$.

5. Let $a > 1$ be a given number. We consider the function

$$f(x) = a^x$$

Using the fact that $f'(0) = \ln a$, find an approximation to $\ln 6$ by using the derivative table in Example 8.4.

6. Repeat the previous exercise to approximate $\ln 3$ to five decimal digits.

7. Given the approximation formula

$$K(x) = \frac{f(x+3h) - f(x-h)}{4h}$$

show that

$$f'(x) - K(x) = c_1 h + c_2 h^2 + \cdots$$

and determine c_1.

8. Using the following data, find $f'(6.0)$, error $= O(h)$, and $f''(6.3)$, error $= O(h^2)$.

x	6.0	6.1	6.2	6.3	6.4
$f(x)$	-0.1750	-0.1998	-0.2223	-0.2422	-0.2596

9. Define

$$G(h) = \frac{-f(x+2h) + 4f(x+h) - 3f(x)}{2h}$$

(a) Show that

$$f'(x) - G(h) = c_1 h^2 + c_2 h^3 + c_3 h^4 + \ldots$$

and state c_1.

(b) Compute $f'(0.398)$ using the table below and $G(h)$.

x	0.398	0.399	0.400	0.401	0.402
$f(x)$	0.408591	0.409671	0.410752	0.411834	0.412915

10. **Computer assignment:**
Write a computer program in FORTRAN or C that finds the first derivative of a function f at a given value α of x, using the central difference approximation and Richardson's formula. Input data

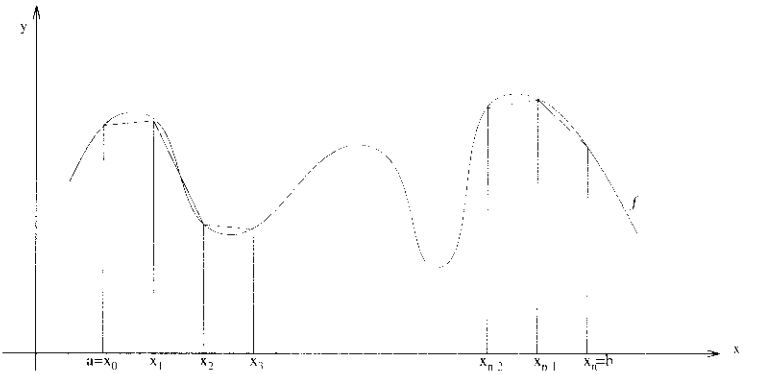


FIGURE 8.1
Trapezoidal rule.

to the program should be a function f, α, the value of h, and the number of rows n.

Test your program to find the derivative of $f(x) = e^{2x}\sin x$ at $x = 0.1, 0.5, 1, 1.5$. Compare the result with the exact answer.

8.3 TRAPEZOIDAL RULE

One of the simplest methods of finding the area under a curve, known as the **trapezoidal rule**, is based on approximating $f(x)$ by a piecewise linear polynomial which interpolates $f(x)$ at the nodes $x_0, x_1, \ldots, x_n$.

In general, suppose we wish to evaluate the integral

$$I = \int_a^b f(x)dx.$$

We begin by dividing the interval $[a, b]$ into n subintervals, each with size h as shown in Figure 8.1, where

$$h = \frac{b-a}{n} \quad \text{and} \quad x_i = a + ih \quad \text{for } i = 0, 1, 2, \ldots, n.$$

We now consider each subinterval $[x_{i-1}, x_i]$, $i = 1, 2, \ldots, n$ and approximate the area under the curve $f(x)$ by a trapezoid with heights $f(x_{i-1})$ and $f(x_i)$. The area of the ith trapezoid is therefore

$$A_i = \frac{h}{2}\left[f(x_{i-1}) + f(x_i)\right] \tag{8.19}$$

The total area T_n is obtained by extending (8.19) over the entire interval $[a, b]$, that is

$$\begin{aligned} T_n &= A_1 + A_2 + \cdots + A_{n-1} + A_n \\ &= \frac{h}{2}\left[f(x_0) + f(x_1)\right] + \frac{h}{2}\left[f(x_1) + f(x_2)\right] + \cdots \\ &\quad + \frac{h}{2}\left[f(x_{n-2}) + f(x_{n-1})\right] + \frac{h}{2}\left[f(x_{n-1}) + f(x_n)\right]. \end{aligned}$$

By collecting common terms, we obtain the formula

$$T_n = \frac{h}{2}\left[f(x_0) + f(x_n)\right] + h\sum_{i=1}^{n-1} f(x_i). \tag{8.20}$$

Eqn. (8.20) is known as the **composite trapezoidal rule** for n subintervals, so called because it approximates the integral I by the sum of n trapezoids.

EXAMPLE 8.5

Use the composite trapezoidal rule with $n = 2$ to approximate

$$\int_0^3 x^2 e^x dx.$$

Using (8.20) with $n = 2$, we have

$$\begin{aligned} \int_0^3 x^2 e^x dx &\approx \frac{3}{4}[f(0) + f(3)] + \frac{3}{2}f(1.5) \\ &\approx 150.70307 \end{aligned}$$

One should note that formula (8.20) gives an approximation to the integral Eqn. (8.1) with no information on the accuracy of the trapezoidal method. Because of the round-off error, taking a large number of trapezoids will not result in general to a better approximation. Since we usually do not have prior knowledge of the exact answer, it is hard to find the number of trapezoids that give the best approximation. One method often used, is by computing the integral several times with an increasing number of trapezoids, and then stop when the difference between two successive answers is satisfactorily small.

m-function 8.2

The following MATLAB function **trapez.m** computes the definite integral 8.1 using the composite trapezoidal rule. INPUTS are a function *f*, the values of *a* and *b*; the number of trapezoids *n*. The input function *f(x)* should be defined as M-file.

```
function trapez(f,a,b,n)
% Compute the integral of f(x) from a to b using the trapezoid rule
h=(b-a)/n;
disp('_____________________________________________________')
disp(['        i          xi          f(xi)        h=',num2str(h) ])
disp('_____________________________________________________')
S=feval(f,a);
fprintf(' %2.0f %12.4f %14.6f\n',0,a,S);
for i=1:n-1
  x=a+h*i;
  g=feval(f,x);
  S=S+2*g;
  fprintf(' %2.0f %12.4f %14.6f\n',i,x,g);
end
S=S+feval(f,b);
fprintf(' %2.0f %12.4f %14.6f\n',n,b,feval(f,b));
INT=h*S/2;
fprintf('\n        The intergral of f(x) is =%16.8f\n',INT);
```

EXAMPLE 8.6

Use the MATLAB function trapez with $n = 5$ to approximate the integral

$$\int_0^1 (7 + 14x^6)dx = 9.$$

Before calling the function trapez, we first define the MATLAB function f1 as follows:

```
function f=f1(x)
f=7+14*x.^6;
```

The results are shown in Table 8.3. Therefore, the trapezoidal rule

» trapez('f1',0,1,5)

i	xi	f(xi)	h=0.2
0	0.0	7	
1	0.2	7.000896	
2	0.4	7.057344	
3	0.6	7.653184	
4	0.8	10.670016	
5	1.0	21	

The intergral of f(x) is = 9.27628800

Table 8.3 The trapezoid rule for $f(x) = 7 + 14x^6$ with $n = 5$.

estimate with $h = \frac{1-0}{5}$ is

$$\begin{aligned} T_5 &= \frac{1}{10}[f(0) + f(1)] + \frac{1}{5}[f(\frac{1}{5}) + f(\frac{2}{5}) + f(\frac{3}{5}) + f(\frac{4}{5})] \\ &\approx \frac{1}{10}[7 + 21] + \frac{1}{5}[7.000896 + 7.057344 + 7.653184 + 10.670016] \\ &\approx 9.276288 \end{aligned}$$

Truncation Error:

Neglecting the round-off error, let us now study the accuracy of the composite trapezoidal rule by analyzing the error in using it to estimate the integral (8.1), known as the truncation error.

By the fundamental theorem of calculus we have

$$I_i = \int_{x_{i-1}}^{x_i} f(x)dx = F(x_i) - F(x_{i-1}) \tag{8.21}$$

where $F'(x) = f(x)$.
Consider the Taylor series expansion of $F(x)$ about x_i. We assume that $f(x)$ has as many continuous derivatives as may be required.

$$\begin{aligned} F(x_{i-1}) &= F(x_i - h) \\ &= F(x_i) - hF'(xi) + \frac{h^2}{2!}F''(xi) - \frac{h^3}{3!}F'''(x_i) + \cdots \end{aligned}$$

or

$$\begin{aligned} F(x_i) - F(x_{i-1}) &= hF'(x_i) - \frac{h^2}{2!}F''(x_i) + \frac{h^3}{3!}F'''(x_i) - \cdots \\ &= I_i. \end{aligned}$$

Since $F'(x) = f(x)$, we have

$$I_i = hf(x_i) - \frac{h^2}{2!}f'(x_i) + \frac{h^3}{3!}f''(x_i) - \cdots. \tag{8.22}$$

It was found in Section 8.1 that

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \frac{h}{2}f''(x_i) + \cdots \tag{8.23}$$

Substituting (8.23) into (8.22), yields

$$\begin{aligned} I_i &= \frac{h}{2}\left[f(x_i) + f(x_{i-1})\right] - \frac{h^3}{12}f''(x_i) + \cdots \\ &= A_i - \frac{h^3}{12}f''(x_i) + \cdots. \end{aligned}$$

Hence, the error term E_i for the ith trapezoid is

$$\begin{aligned} E_i &= I_i - A_i \\ &= -\frac{h^3}{12}f''(x_i) + \text{ higher order terms.} \end{aligned}$$

For h small, we neglect the higher order terms to get

$$E_i = -\frac{h^3}{12}f''(x_i). \tag{8.24}$$

If $|f''(x)| \leq M$ for all x in the interval $[a, b]$, then

$$|E_i| \leq \frac{h^3}{12}M \tag{8.25}$$

Applying (8.25) over the entire interval, we obtain the total error E_T

$$|E_T| \leq n\frac{h^3}{12}M = (b-a)\frac{h^2}{12}M. \tag{8.26}$$

Hence, in view of (8.24), as we expected, the error term for the composite trapezoidal rule is zero if $f(x)$ is linear; that is, for a first-degree polynomial, the trapezoidal rule gives the exact result.

EXAMPLE 8.7

Determine the number of subintervals n required to approximate

$$\int_0^2 \frac{1}{x+4} dx$$

with an error E_T less than 10^{-4} using the composite trapezoidal rule.

We have

$$|E_T| \leq \frac{b-a}{12} h^2 M \leq 10^{-4}$$

In this example, the integrand is $f(x) = 1/(x+4)$, and $f''(x) = 2(x+4)^{-3}$. The maximum value of $|f''(x)|$ on the interval $[0, 2]$ is $1/32$, and thus, $M = 1/32$. This is used with the above formula to obtain

$$\frac{1}{192} h^2 \leq 10^{-4} \quad \text{or} \quad h \leq 0.13856$$

Since $h = 2/n$, the number of subintervals n required is $n \geq 15$.

EXAMPLE 8.8

Use the composite trapezoidal rule with $n = 1$ to compute the integral

$$\int_1^3 (2x+1) dx.$$

From Table 8.4, we have

$$T_1 = \frac{2}{2}[f(1) + f(3)] = 3 + 7 = 10$$

As expected, the trapezoidal rule gives the exact value of the integral because $f(x)$ is linear.

In Table 8.4 the MATLAB function f2 is defined as follows:

```
function f=f2(x)
f=2*x+1;
```

```
» trapez('f2',1,3,1)

i        xi        f(xi)        h=2
0        1         3
1        3         7

The intergral of f(x) is =    10
```

Table 8.4 The composite trapezoidal rule for $f(x) = 2x + 1$ with $n = 5$.

EXERCISE SET 8.3

1. Use the composite trapezoidal rule to approximate the following integrals for the stated values of n.

   (a) $\int_0^{\pi/4} \sin x dx$, $n = 5$.

   (b) $\int_1^2 \frac{2}{\sqrt{x}} dx$, $n = 4$.

   (c) $\int_0^{\pi/3} \cos(3\cos^2 x) dx$, $n = 7$.

   (d) $\int_0^1 \ln(3 + \sin x) dx$, $n = 5$.

2. Eliminate the singularity at $x = 1$ of the integral

$$I = \int_0^1 \frac{2}{\sqrt{1 - x^2}} dx$$

   by using the transformation $x = \sin u$ and then use the composite trapezoidal rule to approximate I.

3. Approximate the integral

$$I = \int_{-2}^{3} (x^4 - 3x^3 + 2x^2 - 3) dx$$

   using the composite trapezoidal rule with $n = 30$. Compare with the exact value $\frac{175}{12}$.

4. Use the composite trapezoidal rule to approximate the integral

$$I = \int_{0.4}^{0.6} e^{2x} \sin 3x dx$$

and find a bound for the error if possible.

5. Given the table

x	0	0.1	0.2	0.3
$f(x)$	2.72	3.00	3.32	3.67

use the composite trapezoidal rule to find an approximation to the integral

$$\int_0^{0.3} f(x)dx$$

6. Approximate the integral

$$\int_0^2 \ln\left(\frac{e^x + 2}{\cos x + 2}\right) dx$$

using the composite trapezoidal rule with $n = 5, 7$, and 9.

7. Find an expression for the error of the following integral rule

$$\int_a^{a+h} f(x)dx \approx hf(a+h) - \frac{1}{2}h^2 f'(a)$$

8. **Computer assignment:**
Write a computer program in FORTRAN or C to evaluate a definite integral using the composite trapezoidal rule. Input data to the program should be the limits of integration a and b, the integrand $f(x)$, and the number of subintervals n.

Test your program to solve Exercise 6.

8.4 SIMPSON'S RULE

In this section, the partitioning of the interval $[a, b]$ is assumed to be made of an even number n of subintervals of width h. Taking them two at a time, as shown in Figure 8.2, Simpson's rule consists of approximating $f(x)$ by a quadratic polynomial interpolating at x_{i-1}, x_i, and x_{i+1}. Using the Taylor series approach, we shall now try to derive

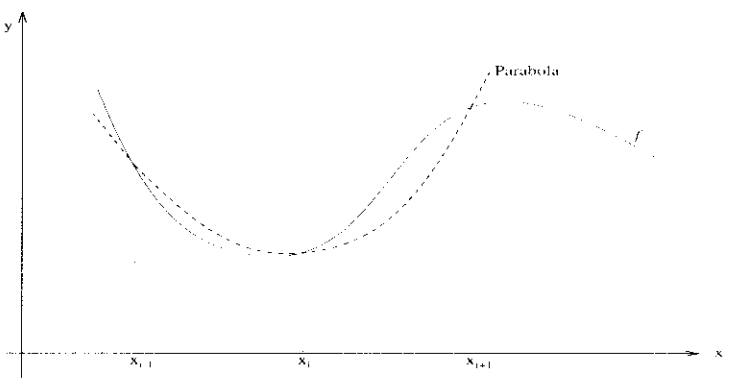


FIGURE 8.2
Simpson's rule.

a formula for Simpson's rule together with the error term.

By the fundamental theorem of calculus, we have

$$I_i = \int_{x_{i-1}}^{x_{i+1}} f(x)dx = F(x_{i+1}) - F(x_{i-1}). \tag{8.27}$$

The Taylor series expansion of $F(x)$ about x_i is

$$\begin{aligned} F(x_{i+1}) &= F(x_i + h) \\ &= F(x_i) + hF'(x_i) + \frac{h^2}{2!}F''(x_i) + \frac{h^3}{3!}F'''(x_i) \\ &\quad + \frac{h^4}{4!}F^{(4)}(x_i) + \frac{h^5}{5!}F^{(5)}(x_i) + \frac{h^6}{6!}F^{(6)}(x_i) + \cdots \\ &= F(x_i) + hf(x_i) + \frac{h^2}{2!}f'(x_i) + \frac{h^3}{3!}f''(x_i) \\ &\quad + \frac{h^4}{4!}f'''(x_i) + \frac{h^5}{5!}f^{(4)}(x_i) + \cdots. \end{aligned} \tag{8.28}$$

Similarly,

$$F(x_{i-1}) = F(x_i) - hf(x_i) + \frac{h^2}{2!}f'(x_i) - \frac{h^3}{3!}f''(x_i) + \frac{h^4}{4!}f'''(x_i) - \frac{h^5}{5!}f^{(4)}(x_i) + \cdots. \tag{8.29}$$

By substituting (8.28) and (8.29) into (8.27), we obtain

$$I_i = 2hf(x_i) + \frac{h^3}{3}f''(x_i) + \frac{h^5}{60}f^{(4)}(x_i) + \cdots. \tag{8.30}$$

It was found in Section 8.1 that

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} - \frac{h^2}{12}f^{(4)}(x_i) + \cdots. \tag{8.31}$$

Finally, Eqns. (8.30) and (8.31) produce

$$I_i = \frac{h}{3}\left[f(x_{i+1}) + 4f(x_i) + f(x_{i-1})\right] - \frac{h^5}{90}f^{(4)}(x_i) + \cdots. \tag{8.32}$$

Hence **Simpson's rule** to approximate the area over two subintervals lying between x_{i-1}, x_i and x_{i+1} is given by

$$S_i = \frac{h}{3}\left[f(x_{i+1}) + 4f(x_i) + f(x_{i-1})\right]. \tag{8.33}$$

We can rewrite Simpson's rule over the interval $[x_0, x_2]$ as

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}\left[f(x_0) + 4f(x_1) + f(x_2)\right] - \frac{h^5}{90}f^{(4)}(\xi) \tag{8.34}$$

where $x_0 < \xi < x_2$. In view of Eqns. (8.32) and (8.34), the error term for Simpson's rule is given by

$$E_i = -\frac{h^5}{90}f^{(4)}(x_i).$$

Again, if $|f^{(4)}(x_i)| \leq M$ for all x in $[a, b]$, then

$$|E_i| \leq \frac{h^5}{90}M. \tag{8.35}$$

EXAMPLE 8.9

Use Simpson's rule to find an approximation to

$$\int_0^3 x^2 e^x dx.$$

Using (8.34), we have

$$\begin{aligned}\int_0^3 x^2 e^x dx &\approx \frac{1}{2}[f(0) + 4f(1.5) + f(3)] \\ &\approx 110.55252\end{aligned}$$

Simpson's composite rule

Formula (8.33) gives the approximation for the integral

$$I = \int_a^b f(x)dx$$

over two equal subintervals. The estimate obtained from this formula will usually not be accurate, particularly when dealing with a large interval $[a, b]$. To obtain a better accuracy, the procedure is extended to n subintervals and Simpson's rule is applied on each pair of consecutive subintervals. Since each application of Simpson's rule requires two subintervals, we assume that $n = 2m$ for some positive integer m. Therefore, by the rule of integration, we have

$$I = \sum_{i=1}^{m} \int_{x_{2i-2}}^{x_{2i}} f(x)dx.$$

Using (8.34), we get

$$I = \frac{h}{3}\sum_{i=1}^{m}[f(x_{2i-2}) + 4f(x_{2i-1}) + f(x_{2i})] - \frac{h^5}{90}\sum_{i=1}^{m} f^{(4)}(\xi_i). \qquad (8.36)$$

This reduces to

$$I = \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{m-1} f(x_{2i}) + 4\sum_{i=1}^{m} f(x_{2i-1}) + f(x_{2m})\right] - \frac{h^5}{90}\sum_{i=1}^{m} f^{(4)}(\xi_i).$$

By neglecting the error term, we get

$$S_n = \frac{h}{3}\left[f(x_0) + 2\sum_{i=1}^{m-1} f(x_{2i}) + 4\sum_{i=1}^{m} f(x_{2i-1}) + f(x_{2m})\right] \qquad (8.37)$$

with

$$h = \frac{b-a}{n}, \quad x_i = a + ih, \quad i = 0, 1, ..., n.$$

Formula (8.37) is called **Simpson's composite rule.**

If (8.35) is applied over n subintervals of $[a, b]$, the error term E_T for the Simpson's composite rule can be expressed as

$$|E_T| \leq \frac{n}{2}\frac{h^5}{90}M = (b-a)\frac{h^4}{180}M. \tag{8.38}$$

Hence, the truncation error bound for Simpson's composite rule is proportional to h^4 whereas the error bound for the composite trapezoidal rule is proportional to h^2. As a result, Simpson's composite rule is more accurate than the composite trapezoidal rule, provided that the round-off error will not cause a problem. Since the error term in (8.36) involves the fourth derivative of f, Simpson's composite rule will give the exact results when applied to any polynomial of degree three or less.

EXAMPLE 8.10

Determine the number of subintervals n required to approximate

$$\int_0^2 \frac{1}{x+4}dx$$

with an error E_T less than 10^{-4} using Simpson's composite rule.

We have

$$|E_T| \leq \frac{b-a}{180}h^4 M \leq 10^{-4}$$

In this example the integrand is $f(x) = 1/(x+4)$, and $f^{(4)}(x) = 24(x+4)^{-5}$. The maximum value of $|f^{(4)}(x)|$ on the interval $[0, 2]$ is $3/128$, and thus $M = 3/128$. This is used with the above formula to obtain

$$\frac{3}{90 \times 128}h^4 \leq 10^{-4} \quad \text{or} \quad h \leq \frac{2}{5}\sqrt[4]{15}$$

Since $h = 2/(2m) = 2/n$, the number of subintervals n required is $n \geq 4$.

EXAMPLE 8.11

Use Simpson's composite rule with $n = 6$ to approximate the integral

$$\int_0^1 (7 + 14x^6)dx = 9.$$

---

m-function 8.3

The following MATLAB function **simpson.m** computes the definite integral 8.1 using Simpson's composite rule. INPUTS are a function *f*; the values of *a* and *b*; the value of *n* (even). The input function *f(x)* should be defined as M-file.

```
function simpson(f,a,b,n)
% Compute the integral of a f from a to b using Simpson's
% composite rule. n must be even.
if n/2~=floor(n/2)
  disp(' n must be even')
  break
end;
h=(b-a)/n;
disp('_____________________________________________________')
disp(['    i            xi              f(xi)            h=',num2str(h) ])
disp('_____________________________________________________')
S=feval(f,a);
fprintf(' %2.0f %12.4f %14.6f\n',0,a,S);
for i=1:n/2
  m=2*i-1;
  x=a+h*m;
  g=feval(f,x);
  S=S+4*g;
  fprintf(' %2.0f %12.4f %14.6f\n',m,x,g);
  m=2*i;
  x=a+h*m;
  g=feval(f,x);
  if(i==n/2)
    S=S+g;
  else
    S=S+2*g;
  end;
  fprintf(' %2.0f %12.4f %14.6f\n',m,x,g);
end
INT=h*S/3;
fprintf('\n        The intergral of f(x) is =%16.8f\n',INT);
```

From the values shown in Table 8.5, Simpson's composite rule estimate with $h = (1-0)/6$ is

$$\begin{aligned} S_6 &= \frac{1}{18}\{f(0) + 2[f(\frac{1}{3}) + f(\frac{2}{3})] + 4[f(\frac{1}{6}) + f(\frac{1}{2}) + f(\frac{5}{6})] + f(1)\} \\ &= \frac{1}{18}[7 + 2(7.019204 + 8.229081) + 4(7.000300 + 7.218750 + \\ &\quad +11.688572) + 21] \\ &= 9.007059 \end{aligned}$$

```
» simpson('f1',0,1,6)
```

i	xi	f(xi)	h=0.16667
0	0	7	
1	0.1667	7.0003	
2	0.3333	7.019204	
3	0.5	7.21875	
4	0.6667	8.229081	
5	0.8333	11.688572	
6	1	21	
The intergral of f(x) is =		9.00705876	

Table 8.5 Simpson's composite rule for $f(x) = 7 + 14x^6$ with $n = 6$.

EXAMPLE 8.12

Use Simpson's composite rule with $n = 2$ to approximate the integral

$$\int_1^2 (4x^3 - 2x + 3)dx.$$

---

we have

$$S_2 = \frac{1}{6}[f(1) + 4f(\frac{3}{2}) + f(2)] = 15$$

As expected, Simpson's rule gives the exact value of the integral due to the fact that $f(x)$ has degree three.

```
» simpson('f3',1,2,2)
```

	xi	f(xi)	h=0.5
0	1.0	5.0	
1	1.5	13.5	
2	2.0	31.0	

The intergral of f(x) is = 15.0

Table 8.6 Simpson's composite rule for $f(x) = 4x^3 - 2x + 3$ with $n = 2$.

EXERCISE SET 8.4

1. Approximate the integrals given in Exercise 1 of Section 8.3 using Simpson's composite rule.

2. Evaluate the following integral by Simpson's rule:

$$I = \int_1^2 (2x^5 + 5x^3 - 3x + 1)dx$$

3. Use Simpson's composite rule to approximate the integral

$$I = \int_0^3 x^2 \cos(x^2 - 1)dx$$

with an accuracy of 10^{-3}.

4. How small must be h in order to compute the integral

$$I = \int_{0.2}^{1.1} e^{2x} dx$$

by Simpson's composite rule with an accuracy of 10^{-4}?

5. Estimate the error in the approximation of the integral

$$\int_2^3 \frac{2}{x}\, dx$$

by Simpson's composite rule with $n = 6$.

6. Use Simpson's composite rule to show that

$$\int_1^{1.6} \frac{2}{x}\, dx < 1 < \int_1^{1.7} \frac{2}{x}\, dx$$

7. Use Simpson's rule to estimate

$$\int_1^2 f(x)dx$$

from the following table:

x	1	1.5	2
$f(x)$	0.5	0.4	0.3

8. Let f be defined by

$$f(x) = \begin{cases} x^2 - x + 1 & \text{if } 0 \le x \le 1 \\ 2x - 1 & \text{if } 1 \le x \le 2 \end{cases}$$

(a) Determine whether f is continuous on $[0, 2]$.

(b) Approximate the integral

$$I = \int_0^2 f(x)dx \quad \text{with } n = 2$$

i. using the composite trapezoidal rule on the interval $[0, 2]$.

ii. using Simpson's composite rule first over $[0, 1]$ and then the composite trapezoidal rule over $[1, 2]$.

iii. using Simpson's composite rule over $[0, 2]$.

9. Compute the integral

$$\int_0^1 \frac{x^p}{x^3 + 12} dx$$

for $p = 0, 2$ using composite trapezoidal and Simpson's composite rule with $n = 2, 8$.

10. Approximate

$$\int_{\pi/4}^{\pi/2} \frac{\cos x \log(\sin x)}{\sin^2 x + 1} dx$$

using Simpson's rule with $n = 8$.

11. **Computer assignment:**
Write a computer program in FORTRAN or C to evaluate a definite integral using Simpson's composite rule. Input data to the program should be the limits of integration, the integrand $f(x)$, and the number of subintervals n.

Test your program to solve Exercise 3.

8.5 ROMBERG ALGORITHM

We now describe another powerful and popular method known as the **Romberg quadrature**, which is based on the use of the composite trapezoidal rule combined with the Richardson extrapolation.

We start out with the composite trapezoidal rule approximation

$$T_n = \frac{h}{2}[f(a) + f(b)] + h\sum_{i=1}^{n-1} f(x_i)$$

to the number I.

Here n represents the number of trapezoids related to h by

$$h = \frac{b-a}{n}$$

and

$$x_i = a + ih, \quad i = 0, 1, \ldots, n.$$

For simplicity, we will consider n to be some power of 2, that is $n = 2^{k-1}$, $k = 1, 2, \ldots$. Hence, n can be viewed as the number of times the interval $[a, b]$ has been halved to produce subintervals of length $h = (b-a)/2^{k-1}$ (see Figure 8.3).

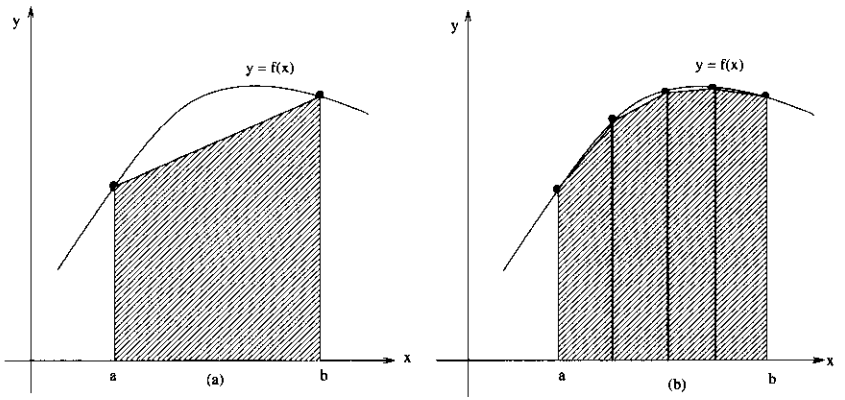


FIGURE 8.3
(a) $R_{1,1}$ = area under $2^0 = 1$ trapezoid. (b) $R_{3,1}$ = area under $2^2 = 4$ trapezoids.

To begin the presentation of the Romberg integration scheme, we adopt the following new notation for the composite trapezoidal rule

$$R_{k,1} = \frac{b-a}{2^k}\left[f(a) + f(b)\right] + \frac{b-a}{2^{k-1}} \sum_{i=1}^{2^{k-1}-1} f\left(a + \frac{b-a}{2^{k-1}} i\right), \quad k = 1, 2, \ldots$$

obtained when the composite trapezoidal rule is applied to 2^{k-1} subintervals; that is, replacing h by $(b-a)/2^{k-1}$ and n by 2^{k-1} in formula (8.20).

Hence,

$$\begin{aligned}
R_{1,1} &= \frac{b-a}{2}\left[f(a) + f(b)\right] \\
R_{2,1} &= \frac{b-a}{4}\left[f(a) + f(b)\right] + \frac{b-a}{2} f\left(a + \frac{b-a}{2}\right) \\
R_{3,1} &= \frac{b-a}{8}\left[f(a) + f(b)\right] + \frac{b-a}{4} \sum_{i=1}^{3} f\left(a + \frac{b-a}{4} i\right) \\
&\text{etc...}
\end{aligned}$$

Note that,

$$R_{2,1} = \frac{R_{1,1}}{2} + \frac{b-a}{2} f\left(a + \frac{b-a}{2}\right)$$

$$R_{3,1} = \frac{R_{2,1}}{2} + \frac{(b-a)}{4}\sum_{i=1}^{2} f\left[a + \frac{b-a}{2}\left(i - \frac{1}{2}\right)\right]$$

etc...

By induction, the general recursion relation for $R_{k,1}$ in terms of $R_{k-1,1}$ is

$$R_{k,1} = \frac{R_{k-1,1}}{2} + \frac{b-a}{2^{k-1}}\sum_{i=1}^{2^{k-2}} f\left[a + \frac{b-a}{2^{k-2}}\left(i - \frac{1}{2}\right)\right] \tag{8.39}$$

For each $k = 2, 3, \ldots, n$.

The recursion relation (8.39) can be used to compute the sequence $R_{2,1}$, $R_{3,1}, \ldots, R_{n,1}$, once $R_{1,1}$ has been calculated.

Assuming that no round-off error enters into calculation, the Richardson extrapolation technique can now be applied to Eqn. (8.39) to improve the estimate of integral I.

It can be shown that the error in the composite trapezoidal rule over 2^{n-1} subintervals can be expressed as

$$C(h^2) + D(h^4) + E(h^6) + \cdots, \quad \text{with} \quad h = \frac{b-a}{2^{n-1}}$$

where $C, D, E \ldots$ are functions of $f(x)$ and its derivatives only.

To proceed with the Richardson extrapolation, consider, for example, the approximation of I using $R_{1,1}$ and $R_{2,1}$

$$I = R_{1,1} + C(h^2) + D(h^4) + \cdots \tag{8.40}$$

$$I = R_{2,1} + C(\frac{h^2}{4}) + D(\frac{h^4}{16}) + \cdots \tag{8.41}$$

Combining (8.40) and (8.41), we get another estimate for I that we will denote by $R_{2,2}$

$$R_{2,2} = \frac{4R_{2,1} - R_{1,1}}{3}$$

This extrapolation eliminates the error term $C(h^2)$.

Similarly, if the extrapolation is carried out for $R_{2,2}$ and $R_{3,2}$, we get a new estimate for I that we will denote by $R_{3,3}$

$$R_{3,3} = \frac{16R_{3,2} - R_{2,2}}{15}$$

This extrapolation eliminates the error term $D(h^4)$.

Following the same procedure we get the general extrapolation formula,

$$R_{i,k} = \frac{4^{k-1}R_{i,k-1} - R_{i-1,k-1}}{4^{k-1} - 1} \tag{8.42}$$

for each $i = 2, \ldots, n$ and $k = 2, \ldots, i$. The truncation error associated with the entry $R_{i,k}$ is of order $O(h^{2k})$.

EXAMPLE 8.13

Compute $R_{2,2}$ to approximate the integral

$$\int_0^3 x^2 e^x dx$$

We have

$$\begin{aligned} R_{1,1} &= \frac{3}{2}[f(0) + f(3)] \\ &= 271.154748 \\ R_{2,1} &= \frac{3}{4}[f(0) + f(3)] + \frac{3}{2}f(\frac{3}{2}) \\ &= 150.703075 \end{aligned}$$

Using (8.42), we get

$$R_{2,2} = \frac{4R_{2,1} - R_{1,1}}{3} = 110.552517.$$

The approximation of I using Eqn. (8.42) is known as the Romberg integral and is often arranged in a form of a diagonal array, as shown in Table 8.7.

Assuming that no round-off error enters into calculations, the extrapolated values along the diagonal will converge to the correct answer more rapidly than the other values below the diagonal. It is interesting to note that the first and second columns in the array contain estimates obtained respectively by the composite trapezoidal and Simpson's rules (see Exercise 7).

$$
\begin{array}{lllll}
R_{1,1} & & & & \\
R_{2,1} & R_{2,2} & & & \\
R_{3,1} & R_{3,2} & R_{3,3} & & \\
R_{4,1} & R_{4,2} & R_{4,3} & R_{4,4} & \\
\vdots & \vdots & \vdots & \vdots & \ddots \\
R_{n,1} & R_{n,2} & R_{n,3} & R_{n,4} & \ldots R_{n,n}
\end{array}
$$

Table 8.7 Romberg array.

EXAMPLE 8.14

Compute the first six rows of the Romberg array to approximate

$$I = \int_0^1 (7 + 14x^6)dx$$

Set

$$f(x) = 7 + 14x^6, \quad a = 0, \quad b = 1 \quad \text{and} \quad n = 6.$$

We start by first computing $R_{1,1}$ and $R_{2,1}$

$$
\begin{aligned}
R_{1,1} &= \frac{1-0}{2}[f(0) + f(1)] = \frac{1}{2}[7 + 21] \\
&= 14 \\
R_{2,1} &= \frac{1-0}{4}[f(0) + f(1)] + \frac{1-0}{2} f(0 + \frac{1-0}{2}) \\
&= 10.609375.
\end{aligned}
$$

Using (8.42), we get

$$
\begin{aligned}
R_{2,2} &= \frac{4R_{2,1} - R_{1,1}}{3} \\
&= \frac{4(10.609375) - 14}{3} = 9.4791666667
\end{aligned}
$$

Next we compute

$$R_{3,1} = \frac{R_{2,1}}{2} + \frac{1}{4}[f(\frac{1}{2} \cdot \frac{1}{2}) + f(\frac{1}{2} \cdot \frac{3}{2})] = 9.4284667969$$

m-function 8.4

The following MATLAB function **romberg.m** computes the definite integral 8.1 using the Romberg integration method. INPUTS are a function *f*; the values *a* and *b*; the number of rows *n*. The input function *f(x)* should be defined as M-file.

```
function romberg(f,a,b,n)
% Compute the integral of f on [a,b] using the Romberg integration.
fprintf('\n')
disp('                Romberg table')
disp('_____________________________________________________')
disp('  i        h        Ri,1        Ri,2        Ri,3      ... ')
disp('_____________________________________________________')
h=b-a;
R(1,1)=h*(feval(f,a)+feval(f,b))/2;
fprintf('%2.0f %8.4f %12.4f\n',1,h,R(1,1));
m=1;
for i=1:n-1
  h=h/2;
  S=0;
  for j=1:m
    x=a+h*(2*j-1);
    S=S+feval(f,x);
  end
  R(i+1,1)=R(i,1)/2+h*S;
  fprintf('%2.0f %8.4f %12.4f',i+1,h,R(i+1,1));
  m=2*m;
  for k=1:i
    R(i+1,k+1)=R(i+1,k)+(R(i+1,k)-R(i,k))/(4^k-1);
    fprintf('%12.4f',R(i+1,k+1));
  end
  fprintf('\n');
end
```

so,

$$R_{3,2} = \frac{4R_{3,1} - R_{2,1}}{3} = 9.0348307292$$

$$R_{3,3} = \frac{16R_{3,2} - R_{2,2}}{15} = 9.0052083333$$

Continuing in this manner leads to the values in Table 8.8.

```
» romberg(TF,0,1,6)
```

Romberg Table

i	h	Ri,1	Ri,2	Ri,3	...		
1	1.0000	14.0000					
2	0.5000	10.6094	9.4792				
3	0.2500	9.4285	9.0348	9.0052			
4	0.1250	9.1088	9.0023	9.0001	9.0000		
5	0.0625	9.0273	9.0001	9.0000	9.0000	9.0000	
6	0.0313	9.0068	9.0000	9.0000	9.0000	9.0000	9.00000000

Table 8.8 The Romberg array for $f(x) = 7 + 14x^6$ with six rows.

EXERCISE SET 8.5

1. Use the Romberg integration with $n = 4$ to approximate the integral

$$\int_1^3 \frac{1}{x}\,dx$$

Compare with the exact value $\ln 3$.

2. Use Romberg integration to compute $R_{4,4}$ for the following integrals:

   (a) $\int_0^{\pi/3} x \sin x dx$.

   (b) $\int_{\pi/7}^{\pi/5} \sin x dx$.

3. Use Romberg integration to approximate

$$\int_0^1 xe^{x^2} dx$$

Complete the table until $|R_{n,n-1} - R_{n,n}| < 10^{-3}$.

4. Approximate $\int_0^2 x^2\sqrt{1+x^2}dx$ using Romberg integration with $n = 4$.

5. Given $R_{4,2} = 5$ and $R_{3,2} = -2$, find $R_{4,3}$ using formula (8.42).

6. Approximate
$$\int_{0.2}^{1.1} \frac{12.1 + \ln(e^{2.4/x} - 1)}{x^4(e^{x^2} + 1)}\, dx$$
using Romberg integration with $n = 6$.

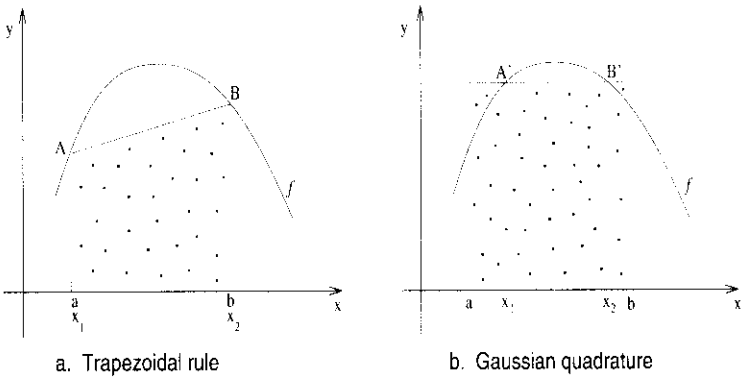
7. Show that the second column in the Romberg array contains estimates obtained by Simpson's rule.

8. **Computer assignment:**
Write a computer program in FORTRAN or C to evaluate a definite integral using the Romberg integral. Input data to the program should be the limits of integration a and b, the integrand $f(x)$, and the number of rows in the Romberg array.

Test your program to solve Exercise 4.

8.6 GAUSSIAN QUADRATURE

All the quadrature formulas developed so far in the preceding sections were of the form
$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{8.43}$$
where w_i are the weights to be given to n functional values $f(x_i)$ and $x_0, x_1, \ldots, x_n$ are the nodes selected to be equally spaced. There are two groups of integration methods which are commonly used: the *Newton-Cotes formulas* that employ equally spaced nodes, and the Gaussian quadrature formulas that employ unequally spaced nodes. The *Gaussian quadrature formula* to be developed in this section has the same form as (8.43), but the nodes to be used are not equally spaced. The reason for choosing unequally spaced nodes is to minimize the error obtained in performing the approximation (8.43) by properly choosing the points x_i. Figure 8.4 shows the difference between the composite trapezoidal rule and the Gauss method with two points. In the composite trapezoidal rule, the approximation of the area under the curve between $x = a$ and $x = b$ is obtained by choosing points x_1 and x_2 at the ends of the interval $[a, b]$. Using the Gaussian quadrature, on the



a. Trapezoidal rule b. Gaussian quadrature

FIGURE 8.4
Comparison of Gaussian quadrature with the trapezoidal rule.

other hand, we choose x_1 and x_2 inside the interval (a, b) so that the area in the trapezoid equals the area under the curve. This is the basic idea of Gaussian rules.

We shall begin first by changing the interval of integration from $[a, b]$ to $[-1, 1]$ using the linear transformation

$$x = a + \frac{b-a}{2}(z+1), \qquad -1 \le z \le 1. \tag{8.44}$$

The corresponding function $g(z)$ is defined by

$$g(z) = f\left(\frac{(b-a)z + (a+b)}{2}\right).$$

Making this substitution in (8.1) gives

$$\int_a^b f(x)dx = \frac{b-a}{2}\int_{-1}^{1} g(z)dz. \tag{8.45}$$

We will now derive the Gaussian quadrature formula for the integral

$$\int_{-1}^{1} g(z)dz.$$

As before, we estimate the integral by approximating the function $g(z)$ with an nth degree interpolating polynomial $p_n(x)$, that is

$$\int_{-1}^{1} g(z) \approx \int_{-1}^{1} p_n(z)dz = \sum_{i=1}^{n} w_i\, g(z_i) \tag{8.46}$$

where the $n+1$ sample points x_i and the weight's factors w_i are to be selected such that the equation

$$\int_{-1}^{1} g(z)dz \approx \sum_{i=1}^{n} w_i \; g(z_i)$$

is exact.

Since the derivation of the Gaussian quadrature formula is too long (see [8]) to present for $n+1$ points, we will derive the formula for three points only. In this case, $g(x)$ is approximated by a parabola that is the degree of the interpolating polynomial $p_n(z)$ is two. The points z_1, z_2, and z_3 and the weight factors w_1, w_2, and w_3 are chosen such that the areas under the curve $g(z)$ and the parabola are equal, that is

$$\text{Area} = w_1 \; g(z_1) + w_2 \; g(z_2) + w_3 \; g(z_3). \tag{8.47}$$

We shall adapt the method of undetermined coefficients to find z_1, z_2, z_3, w_1, w_2, and w_3. Since we have six unknowns, let us assume that the method is exact for the six sample functions

$$\begin{aligned} g(z) &= 1 \\ g(z) &= z \\ g(z) &= z^2 \\ g(z) &= z^3 \\ g(z) &= z^4 \\ g(z) &= z^5 \end{aligned}$$

so that,

$$\begin{aligned} \int_{-1}^{1} 1dz = 2 &= w_1 + w_2 + w_3 \\ \int_{-1}^{1} zdz = 0 &= w_1 z_1 + w_2 z_2 + w_3 z_3 \\ \int_{-1}^{1} z^2 dz = \frac{2}{3} &= w_1 z_1^2 + w_2 z_2^2 + w_3 z_3^3 \\ \int_{-1}^{1} z^3 dz = 0 &= w_1 z_1^3 + w_2 z_2^3 + w_3 z_3^3 \\ \int_{-1}^{1} z^4 dz = \frac{2}{5} &= w_1 z_1^4 + w_2 z_2^4 + w_3 z_3^4 \\ \int_{-1}^{1} z^5 dz = 0 &= w_1 z_1^5 + w_2 z_2^5 + w_3 z_3^5. \end{aligned} \tag{8.48}$$

Observe that (8.48) is a nonlinear system of equations in six unknowns. It has been proved by Gauss that (8.48) has a unique solution given by

$$z_1 = -\sqrt{\frac{3}{5}}, \quad z_2 = 0, \quad z_3 = \sqrt{\frac{3}{5}}$$

$$w_1 = w_3 = \frac{5}{9}, \quad w_2 = \frac{8}{9}.$$

Substituting these constants into (8.46), we obtain the Gaussian three points quadrature formula

$$\int_{-1}^{1} g(z)dz \approx \frac{5}{9}\, g\left(-\sqrt{\frac{3}{5}}\right) + \frac{8}{9}\, g(0) + \frac{5}{9}\, g\left(\sqrt{\frac{3}{5}}\right). \tag{8.49}$$

Since the formula is exact for the polynomial in (8.48), one can show that it will be exact for all polynomials of degree ≤ 5. One should also note that Gauss's method can be extended to four or more points by using the same procedure described above. Values of the appropriate points and weight factors for $n = 2, 3, 4$, and 5 are shown in Table 8.9.

n	Nodes z_i	Weight Factors w_i
2	0.5773502692	1.0
	-0.5773502692	1.0
3	±0.7745966692	0.5555555556
	0.0000000000	0.8888888889
4	±0.3399810436	0.6521451549
	±0.8611363116	0.3478548451
5	±0.5384693101	0.4786286705
	±0.9061798459	0.2369268851
	0.0000000000	0.5688888889
6	±0.2386191861	0.4679139346
	±0.6612093865	0.3607615730
	±0.9324695142	0.1713244924

Table 8.9 Gaussian quadrature nodes and weight factors.

The above derivation applies only if the interval of integration is from -1 to 1. To translate any interval $[a, b]$ into $[-1, 1]$, one should use the transformation (8.44).

EXAMPLE 8.15

Use the Gaussian quadrature formula with $n = 2$ to find an approximation to the integral

$$\int_0^3 x^2 e^x dx.$$

Using the transformation (8.44), we have

$$\int_0^3 x^2 e^x dx = \frac{3}{2}\int_{-1}^{1}\left(\frac{3z+3}{2}\right)^2 e^{\frac{3z+3}{2}} dz$$

Using the values in Table 8.9, we obtain the approximation

$$\begin{aligned}\int_0^3 x^2 e^x &\approx \frac{3}{2}\left[\left(\frac{3\cdot 0.5773502692+3}{2}\right)^2 e^{\frac{3\cdot 0.5773502692+3}{2}}\right.\\ &\quad\left.+\left(\frac{-3\cdot 0.5773502692+3}{2}\right)^2 e^{\frac{-3\cdot 0.5773502692+3}{2}}\right]\\ &\approx 90.60740\end{aligned}$$

EXAMPLE 8.16

Use the Gaussian three-point quadrature formula to evaluate the integral (see Table 8.10)

$$I = \int_0^1 (7 + 14x^6)dx.$$

The three-point Gaussian quadrature formula uses the points 0 and $\pm\sqrt{\frac{3}{5}}$ and the coefficients $\frac{5}{9}$ and $\frac{8}{9}$. This gives, using formula (8.49)

$$\begin{aligned}I &\approx \frac{5}{9}\left[7 + 14\left(\sqrt{\frac{3}{5}}\right)^6\right] + \frac{8}{9}\cdot 7 + \frac{5}{9}\left[7 + 14\left(\sqrt{\frac{3}{5}}\right)^6\right]\\ &\approx 8.995\end{aligned}$$

m-function 8.5

The following MATLAB function **gauss_quad.m** computes the definite integral 8.1 using the Gaussian quadrature formula. INPUTS are a function *f*; the values of *a* and *b*; the value of *n*. The input function *f(x)* should be defined as M-file.

```
function gauss_quad(f,a,b,n)
% Compute the integral of f on [a,b] using Gauss-Legendre
% quadrature.
fprintf('\n')
disp('                Gauss-Legendre quadrature')
disp('_____________________________________________________')
disp('     i       zi          wi          g(zi)        wi*g(zi)    ')
disp('_____________________________________________________')
if (n==2)
  z(1)=-sqrt(1/3);
  z(2)=-z(1);
  w(1)=1; w(2)=1;
end
if (n==3)
  z(1)=-sqrt(3/5);
  z(2)=0;
  z(3)=-z(1);
  w(1)=5/9; w(2)= 8/9; w(3)=w(1);
end
if (n==4)
  z(1)=-sqrt(1/7*(3-4*sqrt(0.3)));
  z(2)=-sqrt(1/7*(3+4*sqrt(0.3)));
  z(3)=-z(1);
  z(4)=-z(2);
  w(1)=1/2+1/12*sqrt(10/3);
  w(2)=1/2-1/12*sqrt(10/3);
  w(3)=w(1); w(4)=w(2);
end
if (n==5)
  z(1)=-sqrt(1/9*(5-2*sqrt(10/7)));
  z(2)=-sqrt(1/9*(5+2*sqrt(10/7)));
  z(3)=0;
  z(4)=-z(1);
  z(5)=-z(2);
  w(1)=0.3*((-0.7+5*sqrt(0.7))/(-2+5*sqrt(0.7)));
  w(2)=0.3*((0.7+5*sqrt(0.7))/(2+5*sqrt(0.7)));
  w(3)=128/225;
```

```
  w(4)=w(1); w(5)=w(2);
end;
S=0;
for i=1:n
  x=((b-a)*z(i)+a+b)/2;
  g=feval(f,x);
  S=S+w(i)*g;
  fprintf('%2.0f %12.4f %12.4f% 12.4f %12.4f\n',i-
1,z(i),w(i),g,g*w(i));
end;
INT=S*(b-a)/2;
fprintf('\n        The intergral of f(x) is =%16.8f\n',INT);
```

This Gaussian three-point quadrature formula gives accuracy comparable to Simpson's rule when used with seven points. This reflects the fact that the Gaussian n-points quadrature formula is exact for polynomials of degree $\leq 2n-1$, which is about twice that of an n-points Newton-Cotes formula.

EXERCISE SET 8.6

1. Use the Gaussian three-point quadrature formula to approximate the following integrals:

   (a) $\int_0^{\pi/4} \sin x dx$.

   (b) $\int_1^2 \frac{2}{\sqrt{x}} dx$.

   (c) $\int_0^{\pi/3} \cos(3\cos^2 x) dx$.

   (d) $\int_0^1 \ln(3+\sin x) dx$.

   Compare your answers with the results obtained in Exercise 1 of Section 8.3.

2. Use the Gaussian quadrature with $n = 2, 3$, and 4 to approximate the integral

$$\int_0^1 \frac{1}{x^2+1} dx$$

```
» gauss('f1',0,1,3)
```

Gauss-Legendre quadrature three-point formula

i	zi	wi	g(zi)	wi*g(zi)
0	0.000000	0.888889	7.218750	6.416667
1	0.774597	0.555556	13.831971	7.684429
2	-0.774597	0.555556	7.000029	3.888905

The intergral of f(x) is = 8.99500000

```
» gauss('f1',0,1,4)
```

Gauss-Legendre quadrature four-point formula

i	zi	wi	g(zi)	wi*g(zi)
0	0.339981	0.652145	8.266310	5.390834
1	-0.339981	0.652145	7.018084	4.576809
2	0.861136	0.347855	16.091115	5.597372
3	-0.861136	0.347855	7.000002	2.434984

The intergral of f(x) is = 9.00000000

Table 8.10 The Gaussian three-point and four-point quadrature formulas.

3. Evaluate the following integral using the Gaussian four-point quadrature:

$$\int_0^{\pi/3} \ln(1 + \cos x)dx$$

4. Use the Gaussian n-points quadrature formula to evaluate the integral

$$\int_{\pi/4}^{\pi/2} (\csc x + 1)(\cot x - 1)dx = -0.90598459.$$

Can you obtain the exact result for some n?

5. Use the Gaussian quadrature with $n = 4$ to approximate

$$\int_0^1 (3x + x^3)e^x dx$$

6. Evaluate
$$\int_1^{1+\sqrt{3}} (2+2x-x^2)^{3/2}dx$$
using the Gaussian quadrature with $n = 4$.

7. Compute the integral
$$\int_{2.1}^{3.1} \frac{\cos 2x}{1+\sin x}dx$$
using the Gaussian quadrature with $n = 2$ and 3.

8. **Computer assignment:**
Write a computer program in FORTRAN or C to evaluate a definite integral using the Gaussian quadrature formula with $n = 2, 3, 4$, or 5. Input data to the program should be the limits of integration, the integrand $f(x)$, the number of points at which the function is to be evaluated, and the nodes and weights corresponding to the different values of n.

   Test your program to solve Exercise 6.

APPLIED PROBLEMS FOR CHAPTER 8

1. The value of π can be estimated by approximating the area of the first quadrant inside the circle $x^2 + y^2 = 1$. The area is equal to 4π. Use the Gaussian quadrature with $n = 4$ to approximate π.

2. The gamma function is defined by
$$\Gamma(\alpha) = \int_0^\infty e^{-x}x^{\alpha-1}dx$$
Approximate $\Gamma(\alpha)$ for values of $\alpha = 1.0$, 1.4, and 2.0.

3. The error function $\text{erf}(x)$ is defined by
$$\text{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$$

Use the Gaussian quadrature with $n = 4$ to approximate erf(0.5) and compare with the tabulated value, erf(0.5) = 0.5200500.

4. An electric dipole consists of opposite charges separated by a small distance d. Suppose that charges of $+q$ and $-q$ units are located on a coordinate line l at $d/2$ and $-d/2$, respectively. The work done in moving the unit charge along l from a to infinity is

$$W(a) = \int_a^\infty \left[\frac{-kq}{(x - d/2)^2} + \frac{kq}{(x + d/2)^2} \right] dx$$

for some positive constant k. Approximate $W(a)$.

5. A simple pendulum consists of a sphere of mass m attached to a string of length L. We assume that the string is weightless and that no other frictional forces are present. If the period T of the pendulum is twice the amount of the time needed for θ to change from $-\theta_0$ to θ_0, then T is given by the improper integral

$$T = 2\sqrt{\frac{2L}{g}} \int_0^{\theta_0} \frac{1}{\sqrt{\cos\theta - \cos\theta_0}} d\theta$$

where g is a gravitational constant and θ is the angle between the vertical and the string. Approximate T if $g = 9.8\, m/s^2$, $\theta_0 = \pi/3$, $L = 2.0\, m$, and $m = 0.5\, kg$.

6. A rod of length l located along the x-axis has a uniform charge per unit length and a total charge Q. The electric potential at a point P along the y-axis at a distance d from the origin is given by

$$V = k\frac{Q}{l} \int_0^l \frac{dx}{\sqrt{x^2 + d^2}}$$

where $k = 9 \times 10^9\, N \cdot m^2/C^2$. Estimate V if $Q = 10^{-6}$ C, $l = 5$ m, and $d = 4$ m.

7. The total emission from an absolute black body is given by the formula

$$E = \int_0^\infty E(v)dv = \frac{2\pi h}{c^3} \int_0^\infty \frac{v^3}{e^{hv/(KT)} - 1} dv$$

Defining $x = hv/(KT)$, we get

$$E = \frac{2\pi h}{c^3} \left(\frac{KT}{h}\right)^4 \int_0^\infty \frac{x^3}{e^x - 1} dx.$$

Approximate the value of the integral correct to six decimal places.

Chapter 9

Numerical Methods for Ordinary Differential Equations

Since the time of Newton, physical phenomena have been investigated by writing them in the form of differential equations. Because the rate of change of a quantity is more easily determined than the quantity itself from a physical experiment, most physical laws of science and engineering are expressed in terms of differential equations. A **differential equation** is an equation that involves at least one derivative of some unknown function.

In this chapter we shall be concerned with numerical methods for solving problems of the form

$$\frac{dy}{dt} = f(t, y) \quad \text{for } a \leq t \leq b \tag{9.1}$$

subject to an initial condition

$$y(a) = y_0.$$

Such problems are called **initial-value problems** (IVPs).

Fortunately, there are many good methods available for solving the initial-value problem (9.1). The methods we shall present here will serve to introduce the basic idea of most other methods and can be easily applied to systems of simultaneous first-order equations.

9.1 EULER'S METHOD

The simplest method of all the numerical methods for ordinary differential equations is **Euler's method**. Although this method is seldom

used, it is relatively easy to analyze and it will illustrate the central ideas involved in the construction of some more advanced methods.

Suppose we want to approximate the solution of the initial-value problem

$$\frac{dy}{dt} = f(t, y), \quad y(a) = y_0 \tag{9.2}$$

on some interval $[a, b]$.

To begin the presentation of Euler's method, let us divide the interval $[a, b]$ into N equal subintervals and define the *mesh points*

$$t_i = a + ih \quad \text{for } i = 0, 1, \ldots, N$$

with $h = (b - a)/N$. h is known as the **step size**. A numerical method for solving the IVP (9.2) will start with the initial condition y_0 and then generates values $y_1, y_2, \ldots, y_N$ which approximate the exact solution $y(t)$ at $t_1, t_2, \ldots, t_N$.

To derive Euler's method, we consider the Taylor expansion of y about t_i for each $i = 0, 1, \ldots, N - 1$. Assuming that $y(t)$ is twice continuously differentiable on $[a, b]$, we get

$$\begin{aligned} y(t_{i+1}) &= y(t_i + h) \\ &= y(t_i) + hy'(t_i) + \frac{h^2}{2} y''(\zeta) \end{aligned} \tag{9.3}$$

for some ζ between t_i and $t_i + h$.

By dropping the error term in Eqn. (9.3) and using Eqn. (9.2), we obtain the formula

$$y(t_i + h) \approx y(t_i) + hf(t_i, y_i)$$

If we denote $y_i \approx y(t_i)$, then

$$y_{i+1} = y_i + hf(t_i, y_i), \quad i = 0, 1, \ldots, N. \tag{9.4}$$

Eqn. (9.4) is known as **Euler's method**.

An algorithm for this method is shown below.

$h \longleftarrow \frac{b-a}{N}$
$y(a) \longleftarrow y_0$
for $i = 0, 1, \ldots, N - 1$
| $y_{i+1} \longleftarrow y_i + hf(t_i, y_i)$

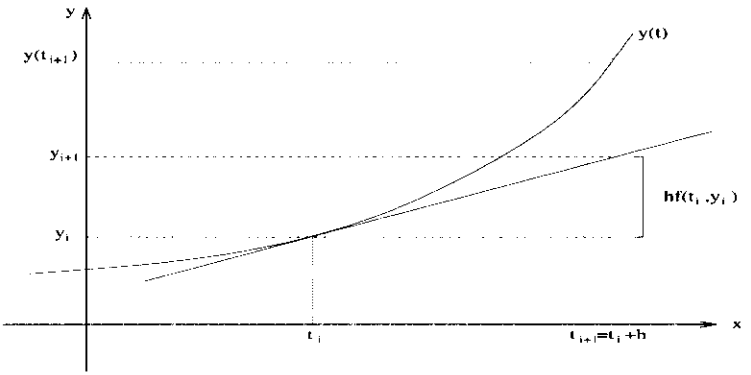


FIGURE 9.1
Geometric interpretation of Euler's method.

A geometric interpretation of this method is given in Figure 9.1 and can be explained as follows:

Suppose we have found y_i at $t = t_i$. The equation of the tangent line to the graph of $y(t)$ at $t = t_i$ is given by

$$y - y_i = m(t - t_i)$$

where $m = \frac{dy}{dt}|_{t=t_i} = f(t_i, y_i)$.

For $t = t_{i+1}$ and $y = y_{i+1}$, we have

$$y_{i+1} - y_i = m(t_{i+1} - t_i).$$

Thus,

$$y_{i+1} = y_i + hf(t_i, y_i).$$

This shows that the next approximation y_{i+1} is obtained at the point where the tangent to the graph of $y(t)$ intersects with the vertical line $t = t_{i+1}$.

EXAMPLE 9.1

Solve the initial-value problem

$$\frac{dy}{dt} = 2t - y, \quad y(0) = -1$$

with $N = 10$ to get the value of y at $t = 1$. Compare with the values of the exact solution $y(t) = e^{-t} + 2t - 2$.

m-function 9.1

The following MATLAB function **euler.m** finds the solution of the initial-value problem 9.2 using Euler's method. INPUTS are a function *f(t,y)*; the initial and final values *a*, *b* of *t*; the initial condition y_0; the number of steps *n;* The input function *f(t,y)* should be defined as an M-file.

```
function euler(f,a,b,y0,n)
% Solve the initial-value problem y'=f(x,y), y(a)=y0
% using Euler's method.
fprintf('\n')
disp('            Euler method')
disp('_____________________________________________________')
disp('   ti        f(ti,yi)          yi          Exact        error     ')
disp('_____________________________________________________')
fprintf('\n')
h=(b-a)/n;
y=y0;
fprintf('%6.2f      ----   %12.6f %12.6f  %4.2f\n',a,y,y,0)
for i=1:n
  t=a+(i-1)*h;
  m=feval(f,t,y);
  y=y+h*m;
% Write the exact solution g if known as g=g(t) otherwise set g='n'.
  t=t+h;
  g='n';
  if (g~='n')
    err=abs(g-y);
    fprintf('%6.2f %12.6f %12.6f %12.6f  %8.2e\n',t,m,y,g,err)
  else
    fprintf('%6.2f %12.6f %12.6f\n',t,m,y)
  end
end
```

With $h = 1/10 = 0.1$ and $f(x, y) = 2t - y$, we have

$$\begin{aligned} y(0.1) \approx y_1 &= y_0 + 0.1 f(0, -1) \\ &= -1 + (0.1)[2(0) - (-1)] \\ &= -0.9 \end{aligned}$$

To calculate $y(0.2)$, the process is repeated starting from the point $(0.1, -0.9)$. The result is

$$\begin{aligned} y(0.2) \approx y_2 &= -0.9 + (0.1)[2(0.1) - (-0.9)] \\ &= -0.79 \end{aligned}$$

Continuing in this manner leads to the values in Table 9.1. At $t = 1.0$ the Euler value is $y_{10} = 0.348678$ and the error is 0.0192.

Note that before calling the MATLAB function euler in Table 9.1, we must define the MATLAB function f1 as follows:

```
function f=f1(t,y)
f=2*t-y;
```

```
>> euler('f1',0,1,-1,10)
```

Euler method

ti	f(t,y)	yi	Exact	Error
0.0	----	-1.000000	-1.000000	0
0.1	1.000000	-0.900000	-0.895163	4.84e-003
0.2	1.100000	-0.790000	-0.781269	8.73e-003
0.3	1.190000	-0.671000	-0.659182	1.18e-002
0.4	1.271000	-0.543900	-0.529680	1.42e-002
0.5	1.343900	-0.409510	-0.393469	1.60e-002
0.6	1.409510	-0.268559	-0.251188	1.74e-002
0.7	1.468559	-0.121703	-0.103415	1.83e-002
0.8	1.521703	0.030467	0.049329	1.89e-002
0.9	1.569533	0.187420	0.206570	1.91e-002
1.0	1.612580	0.348678	0.367879	1.92e-002

Table 9.1 Euler's method for Example 9.1.

EXERCISE SET 9.1

1. Use Euler's method to approximate the solution of the following IVPs in the interval indicated:

   (a) $y' = -2y,\ [0,1],\ y(0) = 4$ with $N = 10$.

   (b) $y' = -\cos(t)y,\ [0,1.2],\ y(0) = 2$ with $N = 12$.

   (c) $y' = -e^t y,\ [0,1],\ y(0) = 3$ with $N = 10$.

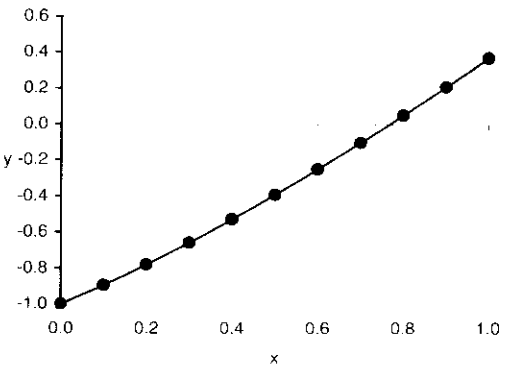


FIGURE 9.2
Euler solution and exact solution $y = e^{-t} + 2t - 2$, $n = 10$.

(d) $y' = -4t^3 y$, $[1, 2]$, $y(1) = 1$ with $N = 10$.

2. Given the initial-value problem

$$y' = -3y\sin(t), \quad \text{with} \quad y(0) = \frac{1}{2}$$

use Euler's method with $N = 10$ to approximate the solution. Compare with the values of the exact solution $y = \frac{1}{2}e^{3(\cos t - 1)}$.

3. Use Euler's method with $h = 0.1$ to approximate the solution of the IVP

$$y' = y - \frac{y}{t}, \quad 1 \le t \le 2, \quad y(1) = \frac{1}{2}.$$

Use the data points generated by Euler's method to find the best function that fits this data in the least squares sense. Use the resulting function to approximate the following values of y:

(a) $y(1.02)$
(b) $y(1.67)$
(c) $y(1.98)$.

Compare with the values of the exact solution $y = \frac{e^t - 1}{2t}$.

4. Given the IVP

$$y' = -\frac{1}{1+t}y, \quad 0 \le t \le 1.6, \quad y(0) = 1$$

with the exact solution $y = \frac{1}{1+t}$:

(a) Use Euler's method to approximate $y(1.6)$ with $N = 8, 16,$ and 32.

(b) Compare your results with the value of the exact solution and find which N gives the best approximation.

5. Given the IVP

$$y' = t - 2ty, \quad 0 \le t \le 1, \quad y(0) = 0$$

with the exact solution $y = \frac{1}{2}(1 - e^{-t^2})$, use Euler's method to approximate the solution with $N = 10$ and compare it with the actual values of y.

6. Use Euler's method to approximate the solution of the following IVPs in the interval indicated:

(a) $y' = -ty$, $[0, 1]$, $y(0) = 2$ with $h = 0.1$.

(b) $y' = -t\sin(t)y$, $[0, 1]$, $y(0) = 1$ with $h = 0.2$.

(c) $y' = \frac{1}{t}y$, $[1, 2]$, $y(1) = 1$ with $h = 0.2$.

(d) $y' = -t\sin(t)y$, $[0, 1]$, $y(0) = 1$ with $h = 0.2$.

(e) $y' = \frac{1+y^2}{ty}$, $[2, 3]$, $y(2) = 3$ with $h = 0.1$.

(f) $y' = \sin t + e^t$, $[0, 5]$, $y(0) = 1$ with $h = 0.5$.

7. **Computer assignment**:
Write a computer program in FORTRAN or C to solve the initial-value problem (9.1) using Euler's method.

Input data to the program should be

(a) The function $f(t, y)$.

(b) The interval $[a, b]$,

(c) The initial value $y(a)$,

(d) The step size h.

Output should consist of the values of y at each step.

Test your program to solve Exercise 2.

9.2 ERROR ANALYSIS

In solving an initial-value problem, it is useful to distinguish two types of errors: *local* and *global* truncation errors. Of course there is always the presence of the round-off error, but we assume here that there is no round-off error involved in any of the calculations.

The **local truncation error** is the error committed in the single step from t_i to t_{i+1}. For Euler's method, the local error is simply the remainder of Taylor's approximation (9.3), that is, $\frac{h^2}{2}y''(\zeta)$. Thus, we see that the local error is $O(h^2)$.

Since at each step of Euler's method an additional truncation error is introduced, the accumulation of these errors is called the **global truncation error**. We will now derive an error bound on the global error for Euler's method. Let us denote the global error after n steps by

$$e_n = y(t_n) - y_n, \quad n = 0, 1, \ldots \tag{9.5}$$

where

$$\begin{aligned} y(t_n) &= \text{value of the true solution at } t = t_n, \\ y_n &= \text{value of the approximate solution at } t = t_n. \end{aligned}$$

We have from Eqn. (9.3)

$$y(t_{n+1}) = y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2}y''(\zeta_n), \quad t_n < \zeta_n < t_{n+1} \tag{9.6}$$

Subtract (9.4) from (9.6) and use (9.5) to get

$$\begin{aligned} e_{n+1} &= e_n + h\,[f(t_n, y(t_n)) - f(t_n, y_n)] + \frac{h^2}{2}y''(\zeta_n) \\ &= e_n + h\,\frac{f(t_n, y(t_n)) - f(t_n, y_n)}{y(t_n) - y_n}\,[y(t_n) - y_n] + \frac{h^2}{2}y''(\zeta_n) \\ &= e_n + h\,\frac{f(t_n, y(t_n)) - f(t_n, y_n)}{y(t_n) - y_n}\,e_n + \frac{h^2}{2}y''(\zeta_n). \end{aligned} \tag{9.7}$$

By assuming continuity on $f(t, y)$ and f_y and using the mean value theorem, we get

$$e_{n+1} = e_n + hf_y(t_n, \xi_n)e_n + \frac{h^2}{2}y''(\zeta_n), \quad y_n < \xi_n < y(t_n). \tag{9.8}$$

Further, if we assume that on the region of interest in the ty-space

$$\left|\frac{\partial}{\partial y}f(t,y)\right| \leq M \quad \text{and} \quad |y''(t)| \leq K$$

where M and K are positive constants, we obtain

$$|e_{n+1}| < (1+hM)|e_n| + \frac{h^2}{2}K. \tag{9.9}$$

We will now show by induction, that

$$|e_n| \leq c[(1+hM)^n - 1], \quad n = 0, 1, ..., \quad c = \frac{hK}{2M}. \tag{9.10}$$

Since $y(t_0) = y_0$, $e_0 = 0$, then, (9.10) is true for $n = 0$. Assuming that (9.10) is true for an integer n, we need to show that it is true for $(n+1)$. We have

$$|e_{n+1}| \leq c[(1+hM)^{n+1} - 1]. \tag{9.11}$$

From (9.9) and (9.10) it follows that

$$\begin{aligned} |e_{n+1}| &\leq c[(1+hM)^n - 1](1+hM) + \frac{h^2}{2}K \\ &= c(1+hM)^{n+1} - c(1+hM) + \frac{h^2}{2}K. \end{aligned}$$

Since $chM = \frac{h^2}{2}K$, simplify the last equation to get (9.11).

Finally, by using Taylor's theorem, it is easy to show that (see Exercise 1)

$$1 + x \leq e^x, \quad \text{for all } x \geq 0.$$

It follows that

$$(1+hM) \leq e^{hM}$$

Hence,

$$(1+hM)^n \leq e^{nhM}$$

Applying this result to Eqn. (9.10), we get

$$|e_n| \leq \frac{hK}{2M}(e^{nhM} - 1) = \frac{hK}{2M}(e^{(t_n - t_0)M} - 1) \tag{9.12}$$

This shows that the global error for Euler's method is $O(h)$. In other words, this means that the error bound is proportional to the step size h.

EXAMPLE 9.2

Consider the initial-value problem

$$y' = y - 1, \quad y(0) = 2, \quad 0 \le t \le 1 \quad \text{with } h = 0.1.$$

We have $f(t, y) = y - 1$ and $|\partial f/\partial y| = 1 = M$.
Since the exact solution is $y = e^t + 1$, we have

$$y'' \le e = K, \quad \text{for all } t \text{ in } [0, 1].$$

Thus, Eqn. (9.12) gives the error bound

$$|e_n| \le \frac{0.1e}{2}(e^{t_n} - 1),$$

using a step size of $h = 0.1$ Table 9.2 gives the actual error, together with the error bound, obtained by using the MATLAB function euler.

In this example, the error bounds turn out to be reasonably sharp. However, the global error bounds are usually hopelessly conservative as can be seen in the following example.

EXAMPLE 9.3

Consider the initial-value problem

$$y' = -y, \quad y(0) = 1, \quad 0 \le t \le 4 \quad \text{with } h = 0.1.$$

We have $f(t, y) = -y$ and $|\partial f/\partial y| = 1 = M$.
The exact solution of this IVP is $y(t) = e^{-t}$. This implies that

$$|y''| = |e^{-t}| \le 1 = K \quad \text{for all } t \text{ in } [0, 4].$$

Thus,

$$|e_n| \le \frac{0.1}{2}(e^{t_n} - 1)$$

t_n	Actual error	Error bound
0.0	0.0	0.0
0.1	0.005171	0.014294
0.2	0.011403	0.030092
0.3	0.018859	0.047551
0.4	0.027725	0.066846
0.5	0.038211	0.088170
0.6	0.050558	0.111738
0.7	0.065036	0.137783
0.8	0.081952	0.166568
0.9	0.101655	0.198381
1.0	0.124539	0.233539

Table 9.2 Comparison of the actual error with the error bound for Example 9.2 using Euler's method with $h = 0.1$.

Table 9.3 lists the actual error together with this error bound. Note that in this example, the error bound gets considerably larger than the actual error as t increases.

EXERCISE SET 9.2

1. Show that
$$0 \le 1 + x \le e^x, \quad \text{for all } x \ge 0$$

2. Let $f(t, y)$ be a function defined on $a \le t \le b$, $c \le y \le d$. We say that $f(t, y)$ satisfies a **Lipschitz condition** with respect to y if there exists a constant L such that
$$|f(t, y_1) - f(t, y_2)| \le L|y_2 - y_1|$$
for all $a \le t \le b$ and $c \le y_1, y_2 \le d$. If f and its partial derivative $\partial f/\partial y$ are continuous, and if
$$\left|\frac{\partial}{\partial y} f(t, y)\right| \le L,$$

t_n	Actual error	Error bound
0.0	0.0	0.0
0.1	0.004837	0.005259
0.2	0.008731	0.011070
0.3	0.011818	0.017493
0.4	0.014220	0.024591
...	...	...
3.5	0.005166	1.605773
3.6	0.004795	1.779912
3.7	0.004448	1.972365
3.8	0.004123	2.185059
3.9	0.003819	2.420122
4.0	0.003535	2.679908

Table 9.3 Comparison of the actual error with the error bound for Example 9.3 using Euler's method with $h = 0.1$.

show that f satisfies a Lipschitz condition on $a \le t \le b$ and $c \le y_1, y_2 \le d$ in the variable y with Lipschitz constant $L =$ constant.
Hint: Apply the mean value theorem to f for fixed t.

3. Show that the function

$$f(t, y) = t|y|$$

satisfies a Lipschitz condition on $[2, 3] \times [1, 2]$.

4. Given the initial-value problem

$$y' = -\frac{1}{1+t}y, \quad 0 \le t \le 1, \quad y(0) = 1$$

with the exact solution $y = \frac{1}{1+t}$, use the latter along with Eqn. (9.12) to obtain an error bound for Euler's method with $h = 0.1$.

5. Given the initial-value problem

$$y' = -y - 2t - 1, \quad 0 \le t \le 1, \quad y(0) = 2$$

with the exact solution $y = e^{-t} - 2t + 1$, find the value of the step size h which produces at least two decimal digits of accuracy at $t = 1$, using Euler's method.

9.3 HIGHER ORDER TAYLOR SERIES METHODS

Euler's method was derived from the Taylor Series using two terms in the series. It should be clear that we can construct an approximate solution of the initial-value problem (9.2) using a larger number of terms. In order to formalize this procedure, we will now derive an approximate solution to our initial-value problem using four terms in the Taylor series. Assuming that the solution $y(t)$ of the IVP (9.2) has four continuous derivatives, we have from the Taylor series expansion of $y(t)$ about $t = t_i$

$$y(t_{i+1}) = y(t_i) + hy'(t_i) + \cdots + \frac{h^3}{3!}y^{(3)}(t_i) + \frac{h^4}{4!}y^{(4)}(\zeta_i) \tag{9.13}$$

with $t_i < \zeta_i < t_{i+1}$.

Neglecting the remainder term in (9.13), we get the difference equation

$$y_{i+1} = y_i + hf(t_i, y_i) + \frac{h^2}{2}f'(t_i, y_i) + \frac{h^3}{6}f''(t_i, y_i) \tag{9.14}$$

where $i = 0, 1, \ldots, N-1$.

By using the chain rule, we obtain

$$\begin{aligned} y' &= f(t, y) \\ y'' &= f' = f_t + f_y y' = f_t + f_y f \\ y''' &= f'' = f_{tt} + 2f_{ty}f + f_{yy}f^2 + f_t f_y + f_y^2 f \end{aligned}$$

Putting the last three equations into Eqn. (9.14) gives the so-called **third-order Taylor's method** formula. It should be clear that continuing in this manner, any derivative of $y(t)$ can be expressed in terms of f and its partial derivatives. However, the practical difficulty of this method is that the various derivatives of y are complicated and in some cases, impossible to find. Even though we have reduced the truncation error by adding more terms in the series, the method is generally impractical.

EXAMPLE 9.4

Use the third-order Taylor's method for the initil-value problem

$$\frac{dy}{dt} = 2t - y, \quad y(0) = -1$$

with $N = 10$ to approximate $y(1)$.

We have

$$f(t, y(t)) = 2t - y$$

The first two derivatives of $f(t, y)$ are

$$f'(t, y) = 2 - y' = 2(1 - t) + y$$
$$f''(t, y) = -2 + y' = 2(t - 1) - y$$

so

$$y_{i+1} = y_i + h\psi_3(t_i, y_i) \tag{9.15}$$

with

$$\begin{aligned}\psi_3(t_i, y_i) &= 2t_i - y_i + \frac{h}{2}[2(1 - t_i) + y_i] + \frac{h^2}{6}[2(t_i - 1) - y_i] \\ &= 2t_i - y_i + \left(\frac{h}{2} - \frac{h^2}{6}\right)[2(1 - t_i) + y_i]\end{aligned}$$

for $i = 0, 1, \ldots, N - 1$.

Since $N = 10$, $h = (1 - 0)/10 = 0.1$, then by substituting the value of h into (9.15), we get the difference equation

$$y_{i+1} = 0.9048333y_i + 0.1903333t_i + 0.0096667$$

with $t_i = 0.1i$ for $i = 0, 1, \ldots, 9$.

The values of the approximate solution y_i are shown in Table 9.4, along with the values of the exact solution $y(t) = e^{-t} + 2t - 2$ and the error.

Note that one could also apply Taylor's method of order k given by the formula

$$y_{i+1} = y_i + h\psi_k(t_i, y_i) \quad \text{for } i = 0, 1, \ldots, N - 1, \tag{9.16}$$

where

$$\psi_k(t_i, y_i) = f(t_i, y_i) + \frac{h}{2!}f'(t_i, y_i) + \cdots + \frac{h^{k-1}}{k!}f^{(k-1)}(t_i, y_i)$$

ti	Taylor's method of order three	Exact solution	Error | y(ti)-yi |
0.00	-1.0	-1.0	0.0
0.10	-0.89516680	-0.89516258	4.22E-06
0.20	-0.78127689	-0.78126925	7.64E-06
0.30	-0.65919216	-0.65918178	1.04E-05
0.40	-0.52969250	-0.52967995	1.25E-05
0.50	-0.39348354	-0.39346934	1.42E-05
0.60	-0.25120380	-0.25118836	1.54E-05
0.70	-0.10343101	-0.10341470	1.63E-05
0.80	0.04931208	0.04932896	1.69E-05
0.90	0.20655245	0.20656966	1.72E-05
1.00	0.36786213	0.36787944	1.73E-05

Table 9.4 $y' = 2t - y$ using third-order Taylor's method with $h = 0.1$.

to obtain a better accuracy to the solution of the initial-value problem (9.2). But as we mentioned before, the calculations of higher derivatives of $f(t, y)$ may become very complicated and time-consuming.

EXERCISE SET 9.3

1. Derive the difference equation corresponding to Taylor's method of order two for the following initial value problems:

   (a) $y' = -2ty^2, \quad 0 \le t \le 1, \quad y(0) = 1.$

   (b) $y' = 3(t-1)^2, \quad 0 \le t \le 1, \quad y(0) = 1.$

2. Use one step of Taylor's method of order two with $h = 0.1$ to calculate an approximate value for $y(0.1)$ of the initial-value problems in Exercise 1.

3. Given the IVP

$$y' = -3 - \frac{2}{t}y, \quad 1 \le t \le 1.4, \quad y(1) = 1$$

   and the exact solution $y(t) = \frac{2}{t^2} - t$, use Taylor's series of order two with $h = 0.2$ to approximate $y(1.4)$. Compare with the actual value.

4. Determine y'', given

(a) $y' = 2t^2y^2$.
(b) $y' = ye^t$.

9.4 RUNGE-KUTTA METHODS

As pointed out earlier, the Taylor methods are generally not practical because of the partial derivatives of $f(t, y)$ which need to be determined analytically. Consequently, Taylor's methods are rarely used. To avoid the differentiation of f, we will now derive methods that produce values y_i of the same accuracy as some of the Taylor methods without the necessity of differentiating $f(t, y)$. We shall now present the simplest of these methods known as **Runge-Kutta method of order 2 (RK2)**.

The form of the formula for this method is obtained by replacing the function $h\psi_k$ in Eqn. (9.16) by the function $ak_1 + bk_2$. The result is

$$y_{i+1} = y_i + ak_1 + bk_2 \tag{9.17}$$

where

$$\begin{aligned} k_1 &= hf(t_i, y_i) \\ k_2 &= hf(t_i + \alpha h, y_i + \beta k_1), \end{aligned}$$

and a, b, α and β are constants to be determined so that (9.17) is as accurate as possible.

To derive these constants, we will make Eqn. (9.17) agree with the Taylor series expansion of $y(t)$ about t_i. We have

$$\begin{aligned} y(t_{i+1}) &= y(t_i) + hy'(t_i) + \frac{h^2}{2}y''(t_i) + \cdots \\ &= y(t_i) + hf(t_i, y(t_i)) + \frac{h^2}{2}f'(t_i, y(t_i)) + \cdots \end{aligned}$$

Since $f'(t_i, y(t_i)) = f_t + f_y f$, it follows that

$$y(t_{i+1}) = y(t_i) + hf + \frac{h^2}{2}(f_t + f_y f) + O(h^3) \tag{9.18}$$

where all the functions in Eqn. (9.18) are evaluated at the point $(t_i, y(t_i))$. We now expand $f(t_i + \alpha h, y_i + \beta k_1)$ in Taylor's series for a function of two variables

$$f(t_i + \alpha h, y_i + \beta k_1) = f(t_i, y_i) + \alpha h f_t + \beta k_1 f_y + O(h^2). \tag{9.19}$$

So, Eqns. (9.17) and (9.19) imply that

$$y(t_{i+1}) = y(t_i) + h(a+b)f + bh^2(\alpha f_t + \beta f f_y) + O(h^3) \tag{9.20}$$

Again, all functions in (9.19) and (9.20) are evaluated at $(t_i, y(t_i))$.

Finally, by comparing (9.18) and (9.20), we obtain the system

$$\begin{cases} a+b=1 \\ \alpha = \beta = \frac{1}{2b} \end{cases} \tag{9.21}$$

which is a system of three equations and four unknowns, and thus one variable can be chosen arbitrarily. One of the commonly used solutions is

$$a = b = \frac{1}{2} \quad \text{and} \quad \alpha = \beta = 1.$$

This leads to the Runge-Kutta method of order 2, sometimes known as the **modified Euler method**

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f\left(t_i + h, y_i + hf(t_i, y_i)\right)\right], \quad i = 0, 1, \ldots, N-1$$

or

$$y_{i+1} = y_i + \frac{h}{2}(k_1 + k_2) \tag{9.22}$$

where

$$k_1 = f(t_i, y_i) \quad \text{and} \quad k_2 = f(t_i + h, y_i + hk_1).$$

Other choices for the parameters are $\alpha = \beta = 1/2$, $a = 0$, $b = 1$. This leads to a formula known as the **midpoint method** given by

$$y_{i+1} = y_i + hf\left(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i)\right), \quad i = 0, 1, \ldots, N-1. \tag{9.23}$$

An algorithm for the modified Euler method is given below.

$h = \frac{a-b}{N}$
$y_0 \leftarrow y(a)$
for $i = 0, 1, ..., N-1$
  $k_1 \leftarrow f(t_i, y_i)$
  $k_2 \leftarrow f(t_i + h, y_i + hk_1)$
  $y_{i+1} \leftarrow y_i + \frac{h}{2}(k_1 + k_2)$

Similarly, an algorithm for the midpoint method is

$$
\begin{array}{ll}
h = \frac{a-b}{N} & \\
y_0 \leftarrow y(a) & \\
\textbf{for} & i = 0, 1, \ldots, N-1 \\
 & k_1 \leftarrow f(t_i, y_i) \\
 & k_2 \leftarrow f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\
 & y_{i+1} \leftarrow y_i + hk_2
\end{array}
$$

Since (9.20) is $O(h^3)$, it follows that the local truncation error for both the midpoint and modified Euler methods is $O(h^3)$. Therefore, both methods are of the second order.

A very popular and often commonly used Runge-Kutta method is the **fourth-order method (RK4)** given by the formula

$$y_{i+1} = y_i + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad \text{for } i = 0, 1, \ldots, N-1 \tag{9.24}$$

with

$$
\begin{aligned}
k_1 &= f(t_i, y_i) \\
k_2 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1) \\
k_3 &= f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2) \\
k_4 &= f(t_i + h, y_i + hk_3)
\end{aligned}
$$

The local truncation error is $O(h^5)$. So, by using the Runge-Kutta method of order four, one can get approximations with accuracies equivalent to Taylor's method of order 4 and without requiring analytic differentiation of $f(t, y)$. To find a bound for the global error for the Runge-Kutta method of order four and higher is difficult (see [21]). However, if the local truncation error is of order $O(h^n)$, it can be shown that the global truncation error is of order $O(h^{n-1})$.

EXAMPLE 9.5

Use the midpoint method to obtain an approximation to the solution of the IVP

$$y' = 2t - y, \quad y(0) = -1$$

at $t = 1$ with $N = 10$.

```
» midpoint('f1',0,1,-1,10)

        Midpoint method
```

ti	k1	k2	yi	Exact	Error
0.0	----	----	-1	-1	0
0.1	1.000000	1.050000	-0.895	-0.895163	1.63e-004
0.2	1.095000	1.140250	-0.780975	-0.781269	2.94e-004
0.3	1.180975	1.221926	-0.658782	-0.659182	3.99e-004
0.4	1.258782	1.295843	-0.529198	-0.52968	4.82e-004
0.5	1.329198	1.362738	-0.392924	-0.393469	5.45e-004
0.6	1.392924	1.423278	-0.250596	-0.251188	5.92e-004
0.7	1.450596	1.478067	-0.10279	-0.103415	6.25e-004
0.8	1.502790	1.527650	0.049975	0.049329	6.46e-004
0.9	1.550025	1.572524	0.207228	0.20657	6.58e-004
1.0	1.592772	1.613134	0.368541	0.367879	6.62e-004

Table 9.5 $y' = 2t - y$ using the midpoint method.

With $h = 1/10 = 0.1$ and $f(x, y) = 2t - y$, the first step of the midpoint method for the approximation of $y(0.1)$ is

$$\begin{aligned} k_1 &= f(0, -1) = 1.0 \\ k_2 &= f[0 + 0.1/2, -1 + 0.1(1/2)] = 1.05 \\ y_1 &= -1 + (0.1)(1.05) = -0.895 \end{aligned}$$

In Table 9.5, the numerical solution is continued and compared with values obtained from the analytical solution. At $t = 1$, the midpoint value is

$$y_{10} = 0.3685409848.$$

EXAMPLE 9.6

Use the Runge-Kutta method of orders 4 and 2 to obtain an approximation to the solution of the IVP

$$y' = 2t - y, \quad y(0) = -1$$

at $t = 1$ with $N = 10$.

m-function 9.2

The following MATLAB function **midpoint.m** finds the solution of the initial-value problem 9.2 using the midpoint method. INPUTS are a function *f(t,y)*; the initial and final values *a*, *b* of *t*; the initial condition y_0; the number of steps *n;* The input function *f(t,y)* should be defined as an M-file.

```
function midpoint(f,a,b,y0,n)
% Solve the initial-value problem y'=f(t,y), y(a)=y0
% using the midpoint method.
fprintf('\n')
disp('          Midpoint method')
disp('_____________________________________________________')
disp('   t        k1        k2          y        Exact        error   ')
disp('_____________________________________________________')
fprintf('\n')
h=(b-a)/n;
y=y0;
m=feval(f,a,y0);
fprintf('%6.2f      ----        ----   %12.6f\n',a,y)
for i=1:n
  t=a+(i-1)*h;
  k1=feval(f,t,y);
  k2=feval(f,t+h/2,y+h*k1/2);
  y=y+h*k2;
  t=t+h;
% Enter the exact solution if known as g=g(t) otherwise set g='n'.
  g='n';
  if (g~='n')
    err=abs(g-y);
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f  %8.2e\n', t, k1, k2,
y, g, err)
  else
    fprintf('%6.2f %12.6f %12.6f %12.6f\n',t,k1,k2,y)
  end
end
```

With $h = 1/10 = 0.1$ and $f(x, y) = 2t - y$, the first step of RK2 for the approximation of $y(0.1)$ is

$$k_1 = f(0, -1) = 1$$
$$k_2 = f(0 + 0.1, -1 + 0.1) = 1.1$$
$$y_1 = -1 + \frac{0.1}{2}(1 + 1.1) = -0.895$$

m-function 9.3

The following MATLAB function **rk2_4.m** finds the solution of the initial-value problem 9.2 using the Runge-Kutta methods. INPUTS are a function *f(t,y)*; the initial and final values *a*, *b* of *t*; the initial condition y_0; the number of steps *n;* the RK order 2 or 4. The input function *f(t,y)* should be defined as an m-file.

```
function rk2_4(f,a,b,y0,n,order)
% solve the initial-value problem y'=f(t,y), y(a)=y0
% using the Runge-Kutta methods.
fprintf('\n')
disp(['      Runge-Kutta method of order = ',num2str(order)])
h=(b-a)/n;
y=y0;
if (order==2)
disp('_____________________________________________________')
disp('   t       k1        k2        y        Exact     error      ')
disp('_____________________________________________________')
fprintf('\n')
fprintf('%6.2f     ----       ----  %12.6f %12.6f  %4.2f\n',a,y,y,0)
  for i=1:n
    t=a+(i-1)*h;
    k1=feval(f,t,y);
    k2=feval(f,t+h,y+h*k1);
    y=y+h*(k1+k2)/2;
    t=t+h;
% Enter the exact solution if known as g=g(t) otherwise set g='n'.
    g='n';
    if (g~='n')
      err=abs(g-y);
      fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f  %8.2e\n',t,k1,k2,
y, g, err)
    else
      fprintf('%6.2f %12.6f %12.6f %12.6f\n',t,k1,k2,y)
    end
  end
end
if (order==4)
disp('_____________________________________________________')
disp('  t     k1      k2      k3      k4        y        Exact     error  ')
disp('_____________________________________________________')
fprintf('\n')
```

```
fprintf('%6.2f      ----      ----      ----      ----  %12.6f %12.6f
%4.2f\n',a,y,y,0)
  for i=1:n
    t=a+(i-1)*h;
    k1=feval(f,t,y);
    k2=feval(f,t+h/2,y+h*k1/2);
    k3=feval(f,t+h/2,y+h*k2/2);
    k4=feval(f,t+h,y+h*k3);
    y=y+h*(k1+2*k2+2*k3+k4)/6;
    t=t+h;
% Enter the exact solution if known as g=g(t) otherwise set g='n'.
    g=exp(-t)+2*t-2;
    if (g~='n')
      err=abs(g-y);
      fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %12.6f %12.6f
%8.2e\n', t, k1, k2, k3, k4, y, g, err)
    else
      fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %12.6f\n',t,k1,k2,
k3, k4, y)
    end
  end
end
```

The first step of RK4 with $h = 0.1$ and $f(t, y) = 2t - y$ is

$$
\begin{aligned}
k_1 &= f(0, -1) = 1 \\
k_2 &= f(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(1)) = 1.05 \\
k_3 &= f(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(1.05)) = 1.0475 \\
k_4 &= f(0 + 0.1, -1 + (0.1)(1.0475)) = 1.09525 \\
y_1 &= -1 + \frac{0.1}{6}[1 + 2(1.05) + 2(1.0475) + 1.09525] = -0.8951625
\end{aligned}
$$

In Tables 9.6 and 9.7, the numerical solutions for RK2 and RK4, respectively, are continued and compared with values obtained from the analytical solution. At $t = 1$ the Runge-Kutta value of order 2 is

$$y_{10} = 0.36785409848$$

and the Runge-Kutta value of order 4 is

$$y_{10} = 0.3678797744.$$

```
» rk2_4('f1',0,1,-1,10,2)

    Runge-Kutta method of order  2
```

ti	k1	k2	yi	Exact	Error
0.0	----	----	-1	-1	0
0.1	1.000000	1.100000	-0.895	-0.895163	1.63e-004
0.2	1.095000	1.185500	-0.780975	-0.781269	2.94e-004
0.3	1.180975	1.262878	-0.658782	-0.659182	3.99e-004
0.4	1.258782	1.332904	-0.529198	-0.52968	4.82e-004
0.5	1.329198	1.396278	-0.392924	-0.393469	5.45e-004
0.6	1.392924	1.453632	-0.250596	-0.251188	5.92e-004
0.7	1.450596	1.505537	-0.10279	-0.103415	6.25e-004
0.8	1.502790	1.552511	0.049975	0.049329	6.46e-004
0.9	1.550025	1.595022	0.207228	0.20657	6.58e-004
1.0	1.592772	1.633495	0.368541	0.367879	6.62e-004

Table 9.6 Runge-Kutta method of order 2 for Example 9.6.

Table 9.8 gives a comparison of the Euler's, midpoint, modified Euler's, and RK4 methods for solving the IVP

$$y' = t + 3\frac{y}{t} \text{ over } [1,2], \quad y(1) = 0 \quad \text{with } h = 0.1.$$

The exact solution to this problem is $y(t) = t^3 - t^2$.

EXERCISE SET 9.4

1. Use the Runge-Kutta method of order 2 to approximate the solution of the following IVPs in the interval indicated:

   (a) $y' = -2y$, $[0,1]$, $y(0) = 4$ with $N = 10$.

   (b) $y' = -\cos(t)y$, $[0,1.2]$, $y(0) = 2$ with $N = 12$.

   (c) $y' = -e^t y$, $[0,1]$, $y(0) = 3$ with $N = 10$.

   (d) $y' = -4t^3 y$, $[1,2]$, $y(1) = 1$ with $N = 10$.

```
» rk2_4k('f1',0,1,-1,10,4)
```

Runge-Kutta method of order 4

ti	k1	k2	k3	k4	yi	Exact	Error
0.0	--	----	----	----	-1	-1	0
0.1	1.000000	1.050000	1.047500	1.095250	-0.895162	-0.895163	8.20e-008
0.2	1.095163	1.140404	1.138142	1.181348	-0.781269	-0.781269	1.48e-007
0.3	1.181269	1.222206	1.220159	1.259253	-0.659182	-0.659182	2.01e-007
0.4	1.259182	1.296222	1.294370	1.329745	-0.52968	-0.52968	2.43e-007
0.5	1.329680	1.363196	1.361520	1.393528	-0.393469	-0.393469	2.75e-007
0.6	1.393469	1.423796	1.422279	1.451241	-0.251188	-0.251188	2.98e-007
0.7	1.451188	1.478629	1.477257	1.503462	0.103414	-0.103415	3.15e-007
0.8	1.503414	1.528244	1.527002	1.550714	0.049329	0.049329	3.26e-007
0.9	1.550671	1.573137	1.572014	1.593469	0.20657	0.20657	3.31e-007
1.0	1.593430	1.613759	1.612742	1.632156	0.36788	0.367879	3.33e-007

Table 9.7 Runge-Kutta method of order 4 for Example 9.6.

2. Repeat Exercise 1 using the Runge-Kutta method of order 4.

3. Given the IVP

$$y' = e^{t^2} - \frac{y}{t}, \quad 1 \le t \le 2, \quad y(1) = \frac{e}{2}$$

use the Runge-Kutta method of order 2 with $N = 10$ to approximate the solution and compare with the values of the exact solution $y = \frac{1}{2t}e^{t^2}$.

4. Use the answers generated in Exercise 3 and the linear least squares method to approximate y at

   (a) $t = 1.25$.

   (b) $t = 1.65$.

   Compare your results with the actual values of y.

5. Use the Runge-Kutta method of order 4 with $h = 0.1$ to approximate the solution of IVP

$$y' = y - \frac{y}{t}, \quad 1 \le t \le 2, \quad y(1) = \frac{1}{2}$$

Use the data points generated by the Runge-Kutta method to find the best function that fits this data in the least squares sense. Use the resulting function to approximate the following values of y:

ti	Exact solution	Euler's method	Error Euler	Midpoint method	Error Midpoint	Mod. Euler's method	Error Mod. Euler	RK4 method	Error RK4
1.0	0	0	0	0	0	0	0	0	0
1.1	0.121	0.100000	2.10E-02	0.119286	1.71E-03	0.118636	2.36E-03	0.120989	1.05E-05
1.2	0.288	0.237273	5.07E-02	0.283995	4.01E-03	0.282438	5.56E-03	0.287977	2.29E-05
1.3	0.507	0.416591	9.04E-02	0.500074	6.93E-03	0.497325	9.67E-03	0.506963	3.74E-05
1.4	0.784	0.642727	1.41E-01	0.773468	1.05E-02	0.769219	1.48E-02	0.783946	5.41E-05
1.5	1.125	0.920455	2.05E-01	1.110124	1.49E-02	1.104040	2.10E-02	1.124927	7.31E-05
1.6	1.536	1.254545	2.81E-01	1.515989	2.00E-02	1.507711	2.83E-02	1.535905	9.46E-05
1.7	2.023	1.649773	3.73E-01	1.997010	2.60E-02	1.986154	3.68E-02	2.022881	1.19E-04
1.8	2.592	2.110909	4.81E-01	2.559133	3.29E-02	2.545290	4.67E-02	2.591854	1.46E-04
1.9	3.249	2.642727	6.06E-01	3.208305	4.07E-02	3.191043	5.80E-02	3.248824	1.76E-04
2.0	4.000	3.250000	7.50E-01	3.950473	4.95E-02	3.929334	7.07E-02	3.999791	2.09E-04

Table 9.8 Comparison of the Euler's, Midpoint, Modified Euler's, and RK4 methods for solving $y' = t + 3\frac{y}{t}$, $y(1) = 0$.

(a) $y(1.02)$

(b) $y(1.67)$

(c) $y(1.98)$

Compare with the values of the exact solution $y = \frac{e^{t-1}}{2t}$.

6. Use the Runge-Kutta method of order 4 to approximate the solution of the following IVPs in the interval indicated:

   (a) $y' = -ty$, $[0, 1]$, $y(0) = 2$ with $h = 0.1$.

   (b) $y' = -t\sin(t)y$, $[0, 1]$, $y(0) = 1$ with $h = 0.2$.

   (c) $y' = \frac{1}{t}y, [1, 3]$, $y(1) = 1$ with $h = 0.2$.

   (d) $y' = \frac{1+y^2}{ty}$, $[2, 3]$, $y(2) = 3$ with $h = 0.1$.

   (e) $y' = \sin t + e^t$, $[0, 5]$, $y(0) = 1$ with $h = 0.5$.

7. When $f(t, y)$ depends only on t, i.e., $f(t, y) = f(t)$, show that the Runge-Kutta method of order 4 reduces to Simpson's rule

$$\int_{t_n}^{t_n+h} f(t)dt = y_{n+1} - y_n$$
$$\approx \frac{h}{6}\left[f(t_n) + 4f(t_n + \frac{h}{2}) + f(t_n + h)\right]$$

8. The exact solution to the IVP

$$y' = y \quad \text{with} \quad y(0) = 1$$

is $y = e^t$. Apply the Runge-Kutta method of order 4 to approximate e^t for $t = 0.5$ and $t = 1.0$, and show that the first five decimal places of the result are correct.

9. Show that the solution of the IVP

$$y' = e^{-x^2}, \quad y(0) = 0$$

is

$$y(x) = \int_0^x e^{-t^2} dt$$

Use the Runge-Kutta method of order 4 with $h = 0.05$ to approximate the value of

$$y(1) = \int_0^1 e^{-t^2} dt.$$

10. **Computer assignment:**
Write a computer program in FORTRAN or C to solve the IVP (9.1) using the Runge-Kutta method of order 4.

Input data to the program should be

(a) The function $f(t, y)$.
(b) The interval $[a, b]$.
(c) The initial value $y(a)$.
(d) The step size h.

Output should consist of the values of y at each time step.

Test your program to solve Exercise 3.

9.5 MULTISTEP METHODS

All the methods discussed so far in the previous sections of this chapter have one thing in common, that is, y_{i+1} was computed knowing only y_i and values of f and its derivatives. So, these are called **single-step methods**. On the other hand, if the knowledge of $y_i, y_{i-1}, \ldots, y_{i-k+1}$ is required for the computation of y_{i+1}, the method is referred to as a

multistep method. One approach to the derivation of these methods is based on the idea of numerical integration and the use of finite-difference methods.

By integrating the solution of the IVP (9.2) between t_i and t_{i+1}, we obtain

$$y(t_{i+1}) - y(t_i) = \int_{t_i}^{t_{i+1}} f(t, y(t))dt.$$

Thus,

$$y(t_{i+1}) = y(t_i) + \int_{t_i}^{t_{i+1}} f(t, y(t))dt. \tag{9.25}$$

To carry out the integration in (9.25) we can use a finite-difference method to approximate $f(t, y)$ at some of the data points $t_0, t_1, \ldots, t_i$. This will lead to the formula

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} p(t)dt \tag{9.26}$$

where $y_i \approx y(t_i)$ and $p(t) \approx f(t, y)$.

9.6 ADAMS-BASHFORTH METHODS

In the Adams-Bashforth method (AB), we suppose that $p(t)$ is given by the Newton backward-difference polynomial $p_{m-1}(t)$, derived in Section 5.2, through m points $(t_i, y(t_i)), \ldots, (t_{i-m+1}, y(t_{i-m+1}))$. Substituting $p_{m-1}(t)$ into Eqn. (9.26), we obtain

$$y_{i+1} = y_i + \int_{t_i}^{t_{i+1}} \sum_{k=0}^{m-1} (-1)^k \binom{-s}{k} \nabla^k f(t_i, y_i)dt.$$

By making the change of variable $s = \frac{t-t_i}{h}$, we obtain

$$y_{i+1} = y_i + h \sum_{k=0}^{m-1} \nabla^k f(t_i, y_i)(-1)^k \int_0^1 \binom{-s}{k} ds \tag{9.27}$$

where

$$\frac{1}{k!h^k} \nabla^k f(t_n, y_n) = f\left[(t_{n-k}, y_{n-k}), \ldots, (t_n, y_n)\right]$$

and

$$\binom{-s}{k} = (-1)^k \frac{s(s+1)\cdots(s+k-1)}{k!}.$$

The coefficients

$$\gamma_k = (-1)^k \int_0^1 \binom{-s}{k} ds$$

can be easily computed from the definition of the binomial coefficient taken as a function of s. As an example, let us calculate γ_k for $k = 0, 1, 2, 3$. We have

$$\gamma_k = \int_0^1 \frac{s(s+1)\cdots(s+k-1)}{k!} ds$$

Thus,

$$\gamma_0 = \int_0^1 ds = 1$$

$$\gamma_1 = \int_0^1 s ds = \frac{1}{2}$$

$$\gamma_2 = \int_0^1 \frac{s(s+1)}{2} ds = \frac{1}{2}\int_0^1 (s^2+s) ds = \frac{5}{12}$$

$$\gamma_3 = \int_0^1 \frac{s(s+1)(s+2)}{6} ds = \frac{1}{6}\int_0^1 (s^3+3s^2+2s) ds = \frac{3}{8}$$

Formula (9.27) is known as **Adams-Bashforth m-step method**. Observe that if $m = 1$ in (9.27), the Adams-Bashforth method leads to Euler's method. A list for the cases $m = 2, 3, 4$, along with the local errors follows:

Adams-Bashforth two-step method (AB2) ($m = 2$):

$$y_{i+1} = y_i + \frac{h}{2}\left[3f(t_i, y_i) - f(t_{i-1}, y_{i-1})\right]. \tag{9.28}$$

$$y_0 = a_0, \quad y_1 = a_1,$$

where $i = 1, 2, \ldots, N-1$. The local truncation error is $O(h^3)$.

Adams-Bashforth three-step method (AB3) ($m = 3$):

$$y_{i+1} = y_i + \frac{h}{12}\left[23f(t_i, y_i) - 16f(t_{i-1}, y_{i-1}) + 5f(t_{i-2}, y_{i-2})\right], \quad (9.29)$$

$$y_0 = a_0, \quad y_1 = a_1, \quad y_2 = a_2,$$

where $i = 2, 3, \ldots, N-1$. The local truncation error is $O(h^4)$.

Adams-Bashforth four-step method (AB4) ($m = 4$):

$$y_{i+1} = y_i + \frac{h}{24}\left[55f(t_i, y_i) - 59f(t_{i-1}, y_{i-1}) + 37f(t_{i-2}, y_{i-2}) \right.$$
$$\left. -9f(t_{i-3}, y_{i-3})\right], \quad (9.30)$$

$$y_0 = a_0, \quad y_1 = a_1, \quad y_2 = a_2, \quad y_3 = a_3,$$

where $i = 3, 4, \ldots, N-1$. The local truncation error is $O(h^5)$.

EXAMPLE 9.7

Use Eqn. (9.27) to derive the Adams-Bashforth three-step method.

We have for $m = 3$,

$$y_{i+1} = y_i + h\left[f(t_i, y_i) + \frac{1}{2}\nabla f(t_i, y_i) + \frac{5}{12}\nabla^2 f(t_i, y_i)\right].$$

Since

$$\begin{aligned}
\nabla f(t_i, y_i) &= hf[(t_{i-1}, y_{i-1}), (t_i, y_i)] \\
&= f(t_i, y_i) - f(t_{i-1}, y_{i-1}) \\
\nabla^2 f(t_i, y_i) &= 2h^2 f[(t_{i-2}, y_{i-2}), (t_{i-1}, y_{i-1}), (t_i, y_i)] \\
&= f(t_i, y_i) - 2f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})
\end{aligned}$$

thus,

$$y_{i+1} = y_i + h\left\{f(t_i, y_i) + \frac{1}{2}[f(t_i, y_i) - f(t_{i-1}, y_{i-1})] + \frac{5}{12}[f(t_i, y_i)\right.$$
$$\left. - 2f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})]\right\}.$$

Simplify the last equation to get (9.29).

The local truncation errors given in (9.28) through (9.30) have been derived using the Newton backward divided-difference formula. For an

example, let us derive the local error of (9.29):

The remainder term R_3 for Newton's backward formula with $m = 3$ is

$$-h^3 f^{(3)}(\zeta)\frac{s(s+1)(s+2)}{3!}.$$

Thus, the local error E_3 for the formula (9.29) is given by

$$E_3 = h\int_0^1 R_3\,ds = h^4 f^{(3)}(\zeta)\frac{1}{6}\int_0^1 [s(s+1)(s+2)]ds.$$

Since the sign of $s(s+1)(s+2)$ does not change on $[0,1]$, it follows that there exists a μ_i between t_{i-2} and t_{i+1} such that

$$E_3 = h^4 f^{(3)}(\mu_i)\frac{3}{8} = h^4 y^{(4)}(\mu_i)\frac{3}{8}. \tag{9.31}$$

Therefore, E_3 is $O(h^4)$. The local error for Adams-Bashforth m-step method takes the form

$$E_m(h) = \gamma_m h^{m+1} y^{m+1}(\mu_i) \quad \text{for some } \mu_i \in (t_{i-m+1}, t_{i+1}).$$

The local errors for (9.28) and (9.30) can be derived similarly.

The major disadvantage common to all multistep methods, in general, and Adams-Bashforth methods, in particular, is that they are not self-starting. Take for instance, the Adams-Bashforth four-step method. Four starting values are needed before the formula can be used. In practice, a single-step method with the same order of accuracy is used to determine the starting values. The Adams-Bashforth of order 4 method is widely used in combination with the Runge-Kutta method of order 4, because both kinds have local errors of $O(h^5)$. The advantage that Adams-Bashforth methods have over single-step methods is that the determination of y_{i+1} requires only one evaluation of $f(t,y)$ per step, whereas Runge-Kutta methods for $n \geq 3$ require four or more function evaluations. For this reason, the multistep methods can be twice as fast as the Runge-Kutta methods of comparable accuracy.

m-function 9.4

The following MATLAB function **abash.m** finds the solution of the initial-value problem 9.2 using Adams-Bashforth methods. INPUTS are a function *f(t,y)*; the initial and final values *a*, *b* of *t*; the initial condition y_0; the number of steps *n;* the AB order 2, 3, or 4. The input function *f(t,y)* should be defined as an M-file.

```
function function abash(f,a,b,y0,n,order)
% solve the initial-value problem y'=f(t,y), y(a)=y0
% using Adams-Bashforth methods.
fprintf('\n Runge-Kutta method gives \n\n')
h=(b-a)/n;
t=(a:h:b+h);
y(1)=y0;
fprintf('y(%2.1f) =%14.8f\n',a,y(1))
  % RK4 to start
for i=1:(order-1)
  k1=feval(f,t(i),y(i));
  k2=feval(f,t(i)+h/2,y(i)+h*k1/2);
  k3=feval(f,t(i)+h/2,y(i)+h*k2/2);
  k4=feval(f,t(i)+h,y(i)+h*k3);
  y(i+1)=y(i)+h*(k1+2*k2+2*k3+k4)/6;
  fprintf('y(%2.1f) =%14.8f\n',t(i)+h,y(i+1))
end
fprintf('\n')
disp(['          Adams-Bashforth method of order ',num2str(order)])
disp('_____________________________________________________')
disp('  t      fi-3      fi-2      fi-1      fi      y      Exact      error ')
disp('_____________________________________________________')
fprintf('\n')
if(order==4)
 % 4th order AB
 for i=order:n
   f1=feval(f,t(i),y(i));
   f2=feval(f,t(i-1),y(i-1));
   f3=feval(f,t(i-2),y(i-2));
   f4=feval(f,t(i-3),y(i-3));
   y(i+1)=y(i)+h*(55*f1-59*f2+37*f3-9*f4)/24;
% Enter the exact solution g if known as g=g(x) otherwise set g='n'.
   x=t(i+1);
   g='n';
   if (g~='n')
     err=abs(g-y(i+1));
```

```
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %12.6f %12.6f
%8.2e\n',t(i)+h,f4,f3,f2,f1,y(i+1),g,err)
   else
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f
%12.6f\n',t(i)+h,f4,f3,f2,f1,y(i+1))
   end
  end
elseif (order==3)
  % 3rd order AB
 for i=order:n
   f1=feval(f,t(i),y(i));
   f2=feval(f,t(i-1),y(i-1));
   f3=feval(f,t(i-2),y(i-2));
   y(i+1)=y(i)+h*(23*f1-16*f2+5*f3)/12;
% Enter the exact solution g if known as g=g(x) otherwise set g='n'.
   x=t(i+1);
   g='n';
   if (g~='n')
    err=abs(g-y(i+1));
    fprintf('%6.2f          %12.6f %12.6f %12.6f %12.6f %12.6f
%8.2e\n',t(i)+h,f3,f2,f1,y(i+1),g,err)
   else
    fprintf('%6.2f          %12.6f %12.6f %12.6f
%12.6f\n',t(i)+h,f3,f2,f1,y(i+1))
   end
 end
else
% 2nd order AB
  for i=order:n
   f1=feval(f,t(i),y(i));
   f2=feval(f,t(i-1),y(i-1));
   y(i+1)=y(i)+h*(3*f1-f2)/2;
% Enter the exact solution g if known as g=g(x) otherwise set g='n'.
   x=t(i+1);
   g='n';
   if (g~='n')
     err=abs(g-y(i+1));
     fprintf('%6.2f        %12.6f %12.6f %12.6f %12.6f  %8.2e\n',
t(i)+h, f2, f1, y(i+1), g, err)
   else
    fprintf('%6.2f        %12.6f %12.6f %12.6f\n',t(i)+h,f2,f1,y(i+1))
   end
  end
end
```

EXAMPLE 9.8

Use the Adams-Bashforth method of orders 2, 3, and 4 to obtain an approximation to the solution of the IVP

$$y' = 2t - y, \quad y(0) = -1$$

with $N = 10$, at $t = 1$. Obtain the starting values from the exact solution $y = e^{-t} + 2t - 2$.

We illustrate the use of Adams-Bashforth of order four to estimate $y(0.4)$. The starting values can be obtained using Runge-Kutta method of order four, but since the exact solution is known, we have

$$y_1 = -0.8951625820, \; y_2 = -0.7812692469, \; y_3 = -0.6591817793$$

and

$$\begin{aligned} f(0.1, -0.8951625820) &= 2(0.1) - (-0.8951625820) = 1.095\,162\,582 \\ f(0.2, -0.7812692469) &= 2(0.2) - (-0.7812692469) = 1.181\,269\,247 \\ f(0.3, -0.6591817793) &= 2(0.3) - (-0.6591817793) = 1.259\,181\,779 \end{aligned}$$

Then, from Eqn. (9.30):

$$\begin{aligned} y_4 &= -0.6591817793 + \frac{0.1}{24}\,[55(1.259\,181\,779) - 59(1.181\,269\,247) \\ &\quad +37(1.095\,162\,582) - 9(1)] \\ y_4 &= -0.5296770801 \end{aligned}$$

A summary of the calculations using Adams-Bashforth of orders 2, 3, and 4 is given in Tables 9.9, 9.10, and 9.11, respectively.

There are many other multistep formulas available for solution of first-order ordinary differential equations, but we will not consider them here. We finish this section by mentioning a multistep method which has the advantage of simplicity and is suitable in many applications,

$$\begin{aligned} y_{i+1} &= y_{i-1} + 2hf(t_i, y_i), \quad \text{for} \quad i = 1, 2, \ldots, N-1 \\ y_0 &= a_0, \quad y_1 = a_1. \end{aligned} \tag{9.32}$$

Eqn. (9.32) is known as the **midpoint predictor** formula and is comparable in simplicity to Euler's method but, has a better local truncation error of order $O(h^3)$.

```
» abash('f1',0,1,-1,10,2)

Runge-kutta method gives

y(0.0) =          -1
y(0.1) =     -0.8951625

        Adams-Bashforth method of order 2
```

ti	fi-3	fi-2	fi-1	fi	yi	Exact	error
0.2			1	1.095163	-0.780888	-0.781269	3.81e-004
0.3			1.095163	1.180888	-0.658513	-0.659182	6.69e-004
0.4			1.180888	1.258513	-0.52878	-0.52968	8.99e-004
0.5			1.258513	1.32878	-0.392389	-0.393469	1.08e-003
0.6			1.32878	1.392389	-0.24997	-0.251188	1.22e-003
0.7			1.392389	1.44997	-0.102094	-0.103415	1.32e-003
0.8			1.44997	1.502094	0.050722	0.049329	1.39e-003
0.9			1.502094	1.549278	0.208009	0.20657	1.44e-003
1.0			1.549278	1.591991	0.369344	0.367879	1.46e-003

Table 9.9 The Adams-Bashforth method of order 2 for Example 9.8.

EXERCISE SET 9.6

1. Use the Adams-Bashforth of order 4 method to approximate the solution of the following IVPs in the interval indicated.

   (a) $y' = -2y$, $[0, 2]$, $y(0) = 4$ with $N = 10$.

   (b) $y' = -\cos(t)y$, $[0, 1.4]$, $y(0) = 2$ with $N = 14$.

   (c) $y' = -e^t y$, $[0, 2]$, $y(0) = 3$ with $N = 10$.

   (d) $y' = -4t^3 y$, $[1, 2]$, $y(1) = 1$ with $N = 10$.

   (e) $y' = -ty$, $[0, 1]$, $y(0) = 2$ with $N = 10$.

   (f) $y' = -t\sin(t)y$, $[0, 2]$, $y(0) = 1$ with $h = 0.2$.

2. Solve the IVP

$$y' = -y + e^{-t}, \quad y(0) = 0, \quad 0 \le t \le 1$$

using the Adams-Bashforth method of order 4 with $N = 10$. Estimate the accuracy of your results. Obtain the starting values from the exact solution $y = te^{-t}$.

```
» abash('f1',0,1,-1,10,3)

Runge-kutta method gives

y(0.0) =          -1
y(0.1) =     -0.8951625
y(0.2) =     -0.7812691

        Adams-Bashforth method of order 3
```

t	fi-3	fi-2	fi-1	fi	y	Exact	error
0.3		1	1.095163	1.181269	-0.659214	-0.659182	3.24e-005
0.4		1.095163	1.181269	1.259214	-0.529736	-0.52968	5.56e-005
0.5		1.181269	1.259214	1.329736	-0.393545	-0.393469	7.59e-005
0.6		1.259214	1.329736	1.393545	-0.25128	-0.251188	9.15e-005
0.7		1.329736	1.393545	1.45128	-0.103518	-0.103415	1.04e-004
0.8		1.393545	1.45128	1.503518	0.049216	0.049329	1.13e-004
0.9		1.45128	1.503518	1.550784	0.206451	0.20657	1.19e-004
1.0		1.503518	1.550784	1.593549	0.367757	0.367879	1.23e-004

Table 9.10 The Adams-Bashforth method of order 3 for Example 9.8.

3. Derive formula (9.32) using the central difference approximation for the first derivative.

4. **Computer assignment:**
   Write a computer program in FORTRAN or C to solve the IVP 9.1 using the Adams-Bashforth method of order 4.

   Input data to the program should be

   (a) The function $f(t, y)$.

   (b) The interval $[a, b]$.

   (c) The initial value $y(a)$.

   (d) The step size h. 2'1

   Output should consist of the values of y at each time step.

   Test your program to solve Exercise 2.

```
» abash('f1',0,1,-1,10,4)

 Runge-kutta method gives

y(0.0) =          -1
y(0.1) =     -0.8951625
y(0.2) =     -0.7812691
y(0.3) =    -0.65918158

        Adams-Bashforth method of order 4
```

ti	fi-3	fi-2	fi-1	fi	yi	Exact	error
0.4	1	1.095163	1.181269	1.259182	-0.529677	-0.52968	3.05e-006
0.5	1.095163	1.181269	1.259182	1.329677	-0.393464	-0.393469	4.98e-006
0.6	1.181269	1.259182	1.329677	1.393464	-0.251181	-0.251188	6.92e-006
0.7	1.259182	1.329677	1.393464	1.451181	-0.103406	-0.103415	8.22e-006
0.8	1.329677	1.393464	1.451181	1.503406	0.049338	0.049329	9.31e-006
0.9	1.393464	1.451181	1.503406	1.550662	0.20658	0.20657	1.01e-005
1.0	1.451181	1.503406	1.550662	1.59342	0.36789	0.367879	1.06e-005

Table 9.11 The Adams-Bashforth method of order 4 for Example 9.8.

9.7 PREDICTOR-CORRECTOR METHODS

In Section 9.6, Adams-Bashforth methods were derived from Eqn. (9.26) using Newton backward-difference polynomial which fits only the previously calculated values $f_i, f_{i-1}, f_{i-2}, \ldots$. These are called **explicit** or **open formulas**. Multistep formulas can be also derived using an interpolating polynomial which fits not only the previously calculated values $f_i, f_{i-1}, \ldots$, but the unknown f_{i+1} as well. These are called **implicit** or **closed formulas**. In order to describe the procedure, we consider the simple formula obtained by replacing the integral in Eqn. (9.26) with the trapezoidal rule (Section 8.3). The result gives the implicit formula

$$y_{i+1} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_{i+1}, y_{i+1})\right] \tag{9.33}$$

where $i = 0, 1, \ldots, N-1$.

A different approach is used to solve (9.33), combining the implicit formula (9.33) and an explicit formula. Suppose, for instance, that a predicted estimate of y_{i+1} has been obtained by using an explicit

method such as Euler's formula. Setting $f_{i+1}^{(0)} = f(t_{i+1}, y_{i+1}^{(0)})$, $f_{i+1}^{(0)}$ can then be used in Eqn. (9.33) to obtain the corrected estimate $y_{i+1}^{(1)}$. To improve, y_{i+1}, $y_{i+1}^{(1)}$ is used to obtain the next corrected estimate $y_{i+1}^{(2)}$. This is continued until convergence occurs to as many digits as desired. The general step of this iteration procedure is given by the formula

$$y_{i+1}^{(n+1)} = y_i + \frac{h}{2}\left[f(t_i, y_i) + f(t_{i+1}, y_{i+1}^{(n)})\right] \tag{9.34}$$

where $n = 0, 1, 2, \ldots$.

A suitable way of terminating the procedure is by using the criterion

$$\left|y_{i+1}^{(n+1)} - y_{i+1}^{(n)}\right| < \epsilon \tag{9.35}$$

where ϵ denotes a specified tolerance.

Experience has shown that the first values $y_{i+1}^{(1)}$ obtained by using one correction of y_{i+1} are as accurate as $y_{i+1}^{(2)}$, $y_{i+1}^{(3)}, \ldots$.

The explicit formula used to predict the approximation $y_{i+1}^{(n)}$ is called a **predictor formula** and the implicit formula (9.34) is called a **corrector formula**. In general, if an explicit and implicit formula are used as a pair of formulas, the explicit fórmula is called the predictor and the implicit one is called the corrector.

9.8 ADAMS-MOULTON METHODS

If the additional interpolating node t_{i+1} is used in the approximation of the integral

$$\int_{t_i}^{t_{i+1}} f(t_i, y(t_i))dt,$$

formula (9.26) becomes

$$y_{i+1} = y_i + h\sum_{k=0}^{m} \nabla^k f(t_{i+1}, y_{i+1})(-1)^k \int_0^1 \binom{1-s}{k} ds \tag{9.36}$$

The case when $m = 3$ leads to the most frequently used formula known as the **fourth-order Adams-Moulton (AM4) formula**,

$$y_{i+1} = y_i + \frac{h}{24}\left[9f(t_{i+1}, y_{i+1}) + 19f(t_i, y_i) - 5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})\right] \tag{9.37}$$

where $i = 2, 3, \ldots, N-1$. The local truncation error is $O(h^5)$.

EXAMPLE 9.9

Derive the fourth-order Adams-Moulton formula using Eqn. (9.36).

Let

$$\gamma'_k = (-1)^k \int_0^1 \binom{1-s}{k} ds.$$

We have,

$$\gamma'_0 = \int_0^1 ds = 1,$$

$$\gamma'_1 = \int_0^1 (1-s)ds = -\frac{1}{2}.$$

$$\gamma'_2 = \int_0^1 (-s)(1-s)ds = -\frac{1}{12}.$$

$$\gamma'_3 = \int_0^1 (-s)(1-s)(-s-1)ds = -\frac{1}{24}.$$

By Substituting these values into (9.36), we obtain

$$\begin{aligned} y_{i+1} &= y_i + h\left[\gamma'_0 f(t_{i+1}, y_{i+1}) + \gamma'_1 \nabla f(t_{i+1}, y_{i+1}) + \gamma'_2 \nabla^2 f(t_{i+1}, y_{i+1}) \right. \\ &\quad \left. + \gamma'_3 \nabla^3 f(t_{i+1}, y_{i+1})\right] \\ &= y_i + h\{f(t_{i+1}, y_{i+1}) - \frac{1}{2}[f(t_{i+1}, y_{i+1}) - f(t_i, y_i)] \\ &\quad - \frac{1}{12}[f(t_{i+1}, y_{i+1}) - 2f(t_i, y_i) + f(t_{i-1}, y_{i-1})] \\ &\quad - \frac{1}{24}[f(t_{i+1}, y_{i+1}) - 3f(t_i, y_i) + 3f(t_{i-1}, y_{i-1}) - f(t_{i-2}, y_{i-2})]\} \end{aligned}$$

Simplify the last equation to get (9.37).

An algorithm for the fourth-order Adams-Moulton method is given

below:

$h = \frac{a-b}{N}$
$y_0 \leftarrow y(a)$
$y_1, y_2 \leftarrow$ determined by using a single-step formula.
for $i = 2, 3, \ldots, N-1$
$y_{i+1}^{(k-1)}$ determined by using an explicit formula.
Compute
$y_{i+1}^{(k)} \leftarrow y_i + \frac{h}{24}[9f(t_{i+1}, y_{i+1}^{(k-1)}) + 19f(t_i, y_i)$
$-5f(t_{i-1}, y_{i-1}) + f(t_{i-2}, y_{i-2})]$ for $k = 1, 2, \ldots$.
Iterate on k until $|y_{i+1}^{(k)} - y_{i+1}^{(k-1)}| < \epsilon$
$y_{i+1} \leftarrow y_{i+1}^{(k)}$

A popular predictor-corrector method uses the fourth-order Adams-Moulton formula as the *corrector* method and the fourth-order Adams-Bashforth formula as the *predictor* method. Since four starting values are needed before the Adams-Bashforth method can be used, a single-step formula such as Runge-Kutta method of order four is frequently employed to meet this need.

One question that can arise is, why should one bother using the predictor-corrector methods when the single-step methods are of comparable accuracy to the predictor-corrector methods of the same order. A partial answer to that relies in the actual number of functional evaluations. Take for example, the Runge-Kutta method of order four. Each step requires four evaluations whereas the Adams-Moulton method of the same order requires only as few as two evaluations. For this reason, predictor-corrector formulas are, in general, considerably more accurate and faster than the single-step ones.

EXAMPLE 9.10

Use Adams-Moulton method of order 4 for the IVP

$$y' = 2t - y, \quad y(0) = -1$$

with $N = 10$ to approximate $y(1)$.

We illustrate the use of Adams-Moulton method of order 4 to estimate $y(0.4)$. The starting values can be obtained using Runge-Kutta method of order 4, that is

$$y_1 = -0.89516250, \; y_2 = -0.78126921, \; y_3 = -0.65918158$$

m-function 9.5

The following MATLAB function **amoulton.m** finds the solution of the initial-value problem 9.2 using the Adams-Moulton method of order 4. INPUTS are a function *f(t,y)*; the initial and final values *a*, *b* of *t*; the initial condition y_0; the number of steps *n;* The input function *f(t,y)* should be defined as an M-file.

```
function amoulton(f,a,b,y0,n)
% Solve the initial value problem y'=f(t,y), y(a)=y0
% using the Adams-Moulton method of order 4.
fprintf('\n Runge-Kutta method of order 4 gives \n\n')
h=(b-a)/n;
t=(a:h:b+h);
y(1)=y0;
fprintf('y(%2.1f) =%14.8f\n',a,y(1))
    % RK4 method to start
for i=1:3
  k1=feval(f,t(i),y(i));
  k2=feval(f,t(i)+h/2,y(i)+h*k1/2);
  k3=feval(f,t(i)+h/2,y(i)+h*k2/2);
  k4=feval(f,t(i)+h,y(i)+h*k3);
  y(i+1)=y(i)+h*(k1+2*k2+2*k3+k4)/6;
  fprintf('y(%2.1f) =%14.8f\n',t(i)+h,y(i+1))
end;
fprintf('\n')
disp('            Adams-Moulton method of order 4')
disp('_____________________________________________________')
disp('  t    fi-2    fi-1    fi    fi+1        y        Exact    error    ')
disp('_____________________________________________________')
fprintf('\n')
for i=4:n
  f1=feval(f,t(i),y(i));
  f2=feval(f,t(i-1),y(i-1));
  f3=feval(f,t(i-2),y(i-2));
  f4=feval(f,t(i-3),y(i-3));
  % AB to predict
  w=y(i)+h*(55*f1-59*f2+37*f3-9*f4)/24;
  % AM to correct
  f5=feval(f,t(i+1),w);
  y(i+1)=y(i)+h*(9*f5+19*f1-5*f2+f3)/24;
% Write the exact solution g if known as g=g(x) otherwise st g='n'.
  x=t(i+1);
  g='n';
```

```
  if (g~='n')
    err=abs(g-y(i+1));
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %12.6f %12.6f
%8.2e\n',t(i)+h,f3,f2,f1,f5,y(i+1),g,err)
  else
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %12.6f\n', t(i)+h, f3,
f2, f1, f5, y(i+1))
  end
end
```

and

$$f(0.1, -0.89516250) = 1.095\,162\,50$$
$$f(0.2, -0.78126921) = 1.181\,269\,21$$
$$f(0.3, -0.65918158) = 1.259\,181\,58$$

Then, the Adams-Bashforth formula of order 4 gives

$$\begin{aligned} y_4 &= -0.65918158 + \frac{0.1}{24}\,[55(1.259\,18158) - 59(1.181\,269\,21) \\ &\quad +37(1.095\,162\,50) - 9(1)] \\ y_4 &= -0.5296769299 \end{aligned}$$

and

$$f(0.4, y_4) = f(0.4, -0.5296769299) = 1.32967693.$$

The corrector formula (9.37) then gives

$$\begin{aligned} y_4 &= -0.65918158 + \frac{0.1}{24}\,[9(1.329\,676\,93) + 19(1.259\,181\,58) \\ &\quad -5(1.181\,269\,21) + (1.095\,162\,50)] \\ y_4 &= -0.529680082 \end{aligned}$$

In Table 9.12, the numerical solution is continued and compared with values obtained from the analytical solution. At $t = 1$, the Adams-Moulton value of order 4 is $y_{10} = 0.367878366$.

```
» amoulton('f1',0,1,-1,10)

Runge-kutta method of order 4 gives

y(0.0) =          -1
y(0.1) =     -0.8951625
y(0.2) =     -0.7812691
y(0.3) =    -0.65918158

        Adams-Moulton method of order 4
```

ti	fi-2	fi-1	fi	fi+1	yi	Exact	error
0.4	1.095163	1.181269	1.259182	1.329677	-0.52968	-0.52968	1.28e-007
0.5	1.181269	1.259182	1.32968	1.393467	-0.39347	-0.393469	3.91e-007
0.6	1.259182	1.32968	1.39347	1.451186	-0.251189	-0.251188	6.04e-007
0.7	1.32968	1.39347	1.451189	1.503413	-0.103415	-0.103415	7.72e-007
0.8	1.39347	1.451189	1.503415	1.55067	0.049328	0.049329	9.04e-007
0.9	1.451189	1.503415	1.550672	1.593429	0.206569	0.20657	1.00e-006
1.0	1.503415	1.550672	1.593431	1.63212	0.367878	0.367879	1.08e-006

Table 9.12 The Adams-Moulton method for Example 9.10.

EXERCISE SET 9.8

1. Repeat Exercise 1 of Section 9.1 using the Adams-Moulton method of order 4.

2. Solve the equation

$$y' = y - \frac{y}{t}, \quad y(1) = \frac{1}{2}, \quad 1 \le t \le 2$$

using the Adams-Moulton method of order 4 with $N = 10$. Compare your results with the values of the exact solution $y = \frac{e^t - 1}{2t}$.

3. **Computer Assignment:**
Write a computer program in FORTRAN or C to solve the IVP (9.2) using the Adams-Moulton method of order 4. Use the Runge-Kutta method of order 4 to determine the starting values.

Input data to the program should be

(a) The function $f(t, y)$.

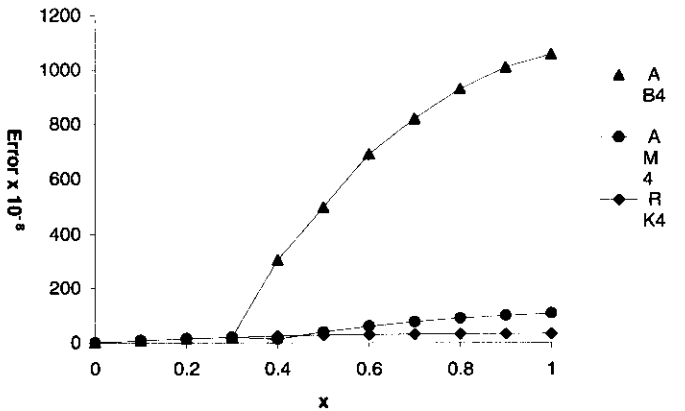


FIGURE 9.3
Errors for the RK4, AB4, and AM4 solutions to the IVP $y' = 2t - y$, $y(0) = -1$, $N = 10$.

(b) The interval $[a, b]$.

(c) The initial value $y(a)$.

(d) The step size h.

Output should consist of the values of y at each time step.

Test your program to solve Exercise 2.

9.9 NUMERICAL STABILITY

Because of round-off error in the computer, errors are committed as the explicit calculation is carried out. Whether these errors amplify or decay, characterizes the stability property of the scheme. A method is said to be **stable** if a small error at one step does not have increasingly great effect on subsequent calculations. In this section, we will show how a method which has the property of being accurate, can produce results that are completely incorrect. To begin our discussion of stability, let us consider the initial-value problem known as the *test*

equation

$$y' = -3y, \quad y(0) = 1. \tag{9.38}$$

The solution of this equation is easily obtained: $y(t) = e^{-3t}$. Problem (9.38) is useful as a test equation for the study of stability of IVP because the exact solution of the test equation has constant relative error as t moves away from y_0. This means that any relative error growth of a numerical solution must be caused by the method that generates the approximate values of the $y(t_i)$'s.

Let us approximate $y(3)$ by using both the Euler and midpoint predictor methods. To use the latter, given by Eqn. (9.32), we need two starting values y_0 and y_1. For y_1 we use the exact value $y(t_1)$. Table 9.13 illustrates the results obtained with $h = 0.1$.

t_i	Exact solution	Euler's method	Error Euler	Midpoint Predictor	Error Midpt. Pred.
0	1	1	0	1	0
0.1	0.740818	0.700000	0.040818	0.740818	0
0.2	0.548812	0.490000	0.058812	0.555509	0.006697
0.3	0.406570	0.343000	0.063570	0.407513	0.000943
0.4	0.301194	0.240100	0.061094	0.311001	0.009807
0.5	0.223130	0.168070	0.055060	0.220912	0.002218
0.6	0.165299	0.117649	0.047650	0.178454	0.013155
0.7	0.122456	0.082354	0.040102	0.113839	0.008617
0.8	0.090718	0.057648	0.033070	0.110151	0.019433
0.9	0.067206	0.040354	0.026852	0.047749	0.019456
1.0	0.049787	0.028248	0.021540	0.081501	0.031714
1.1	0.036883	0.019773	0.017110	-0.001152	0.038035
1.2	0.027324	0.013841	0.013482	0.082192	0.054868
1.3	0.020242	0.009689	0.010553	-0.050467	0.070709
1.4	0.014996	0.006782	0.008213	0.112472	0.097477
1.5	0.011109	0.004748	0.006361	-0.117950	0.129059
1.6	0.008230	0.003323	0.004906	0.183243	0.175013
1.7	0.006097	0.002326	0.003770	-0.227896	0.233993
1.8	0.004517	0.001628	0.002888	0.319980	0.315464
1.9	0.003346	0.001140	0.002206	-0.419884	0.423230
2.0	0.002479	0.000798	0.001681	0.571910	0.569432
2.1	0.001836	0.000559	0.001278	-0.763030	0.764867
2.2	0.001360	0.000391	0.000969	1.029729	1.028368
2.3	0.001008	0.000274	0.000734	-1.380867	1.381875
2.4	0.000747	0.000192	0.000555	1.858249	1.857503
2.5	0.000553	0.000134	0.000419	-2.495817	2.496370
2.6	0.000410	0.000094	0.000316	3.355739	3.355330
2.7	0.000304	0.000066	0.000238	-4.509260	4.509564
2.8	0.000225	0.000046	0.000179	6.061296	6.061071
2.9	0.000167	0.000032	0.000134	-8.146038	8.146204
3.0	0.000123	0.000023	0.000101	10.948918	10.948795

Table 9.13 Comparison of the exact and approximate solution for the IVP $y' = -3y$, $y(0) = 1$ using Euler's and the midpoint predictor methods.

The error columns show that the midpoint predictor method produces a smaller error for small values of t_n than Euler's method. However, as t approaches 3, the results with the midpoint predictor method grow to have very large error, while the results by Euler's method do not show a deviation of the numerical solution from the true solution. This behavior is known as numerical **instability**.

Let us now study the nature of this instability. Applying the midpoint method to our example, gives the difference equation

$$y_{i+1} - y_{i-1} + 6hy_i = 0, \quad y(0) = 1. \tag{9.39}$$

The general solution of (9.39) is

$$y_i = c_1\lambda_1^i + c_2\lambda_2^i \quad \text{where} \quad c_1 + c_2 = 1 \tag{9.40}$$

and λ_1 and λ_2 are the roots of the *characteristic equation*

$$\lambda^2 + 6h\lambda - 1 = 0, \tag{9.41}$$

given by

$$\lambda_1 = -3h + \sqrt{1 + 9h^2} \quad \text{and} \quad \lambda_2 = -3h - \sqrt{1 + 9h^2} \tag{9.42}$$

The binomial approximation of $\sqrt{1 + 9h^2}$ for a small h gives

$$\lambda_1 = -3h + [1 + \frac{9}{2}h^2 + O(h^4)] = e^{-3h} + O(h^3)$$

and

$$\lambda_2 = -3h - [1 + \frac{9}{2}h^2 + O(h^4)] = -e^{3h} + O(h^3).$$

Thus the general solution of (9.40) is

$$y_i = c_1\left(e^{-3h}\right)^i + c_2(-1)^i\left(e^{3h}\right)^i + O(h^3).$$

Since $hi = t_i$ and $c_1 = 1 - c_2$, we have the solution

$$y_i = (1 - c_2)e^{-3t_i} + c_2(-1)^i e^{3t_i} + O(h^3) \tag{9.43}$$

Observe that the first term of (9.43) represents the exact solution of the differential equation and the second is extraneous. It is called the *parasitic* term of the numerical solution because it is unrelated to either the exact solution term $(1 - c_2)e^{-3t_i}$ or the $O(h^3)$ per-step error term.

The exact solution of the problem requires that we choose $c_2 = 0$. However, even a small round-off error may cause c_2 to be nonzero, and the parasitic term will affect the solution unless c_2 is exactly zero. As $t \to \infty$, the true solution $y(t) \to 0$ and the oscillatory term $(-1)^i e^{3t_i}$ will eventually dominate the solution y_i, and the resulting numerical solution will no longer resemble to the exact one. Hence, the midpoint predictor method is an unstable method.

In general, to determine whether a given multistep method is stable, we examine the error produced when the method is applied to the *test equation*

$$y' = \lambda y, \quad y(0) = \alpha, \quad \lambda = \text{negative constant}.$$

This procedure is usually considered sufficient. A stability analysis of single-step or self-starting methods such as Euler's and all Runge-Kutta methods has shown that these methods are stable for sufficiently small h. The stability of a predictor-corrector scheme will depend on both the predictor and corrector formulas but is affected more strongly by the latter if the correction is small.

EXERCISE SET 9.9

1. Show that the general solution of the difference equation obtained by applying the test equation to Milne's corrector formula

$$y_{i+1} = y_{i-1} + \frac{h}{3}[f(t_{i+1}, y_{i+1}) + 4f(t_i, y_i) + f(t_{i-1}, y_{i-1})]$$

is given by

$$y_i = c_1 e^{\lambda t_i} + c_2(-1)^i e^{-\frac{\lambda}{3} t_i}$$

9.10 HIGHER ORDER EQUATIONS AND SYSTEMS OF EQUATIONS

In this section, we present a brief introduction to higher order equations and systems. The methods developed in the previous sections can now be extended easily to solve higher order initial-value problems.

Consider the general nth-order initial-value problem

$$y^{(n)}(t) = f(t, y(t), y'(t), \cdots, y^{(n-1)}(t)), \quad a \le t \le b \tag{9.44}$$

subject to the initial conditions

$$\begin{aligned} y(a) &= \alpha_0, \\ y'(a) &= \alpha_1, \\ &\vdots \\ y^{(n-1)}(a) &= \alpha_{n-1}. \end{aligned}$$

The general approach to solving the initial-value problem (9.44) is to convert it to a system of n first-order equations as follows. Let us define the new variables

$$y_1(t) = y(t),\ y_2(t) = y'(t), \cdots,\ y_n(t) = y^{(n-1)}(t). \tag{9.45}$$

Thus, Problem (9.44) becomes

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ y_{n-1}' &= y_n \\ y_n' &= f(t, y_1, y_2, ..., y_n) \end{aligned} \tag{9.46}$$

with initial conditions

$$y_1(a) = \alpha_0,\ y_2(a) = \alpha_1, \ldots,\ y_n(a) = \alpha_{n-1}.$$

The solution of the system (9.46) can now be carried out by using one of the methods developed in the previous sections of this chapter. As an example, we consider the Runge-Kutta method of order 4 applied to two simultaneous equations of the form

$$\begin{aligned} x' &= f(t, x(t), y(t)), \quad x(a) = \alpha \\ y' &= g(t, x(t), y(t)), \quad y(a) = \beta \end{aligned} \tag{9.47}$$

From the Runge-Kutta formula (9.24), we have

$$\begin{aligned} x_{n+1} &= x_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \\ y_{n+1} &= y_n + \frac{h}{6}(c_1 + 2c_2 + 2c_3 + c_4) \end{aligned} \tag{9.48}$$

where

$$
\begin{aligned}
k_1 &= f(t_n, x_n, y_n) \\
c_1 &= g(t_n, x_n, y_n) \\
k_2 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1, y_n + \frac{h}{2}c_1\right) \\
c_2 &= g\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_1, y_n + \frac{h}{2}c_1\right) \\
k_3 &= f\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2, y_n + \frac{h}{2}c_2\right) \\
c_3 &= g\left(t_n + \frac{h}{2}, x_n + \frac{h}{2}k_2, y_n + \frac{h}{2}c_2\right) \\
k_4 &= f\left(t_n + h, x_n + hk_3, y_n + hc_3\right) \\
c_4 &= g\left(t_n + h, x_n + hk_3, y_n + hc_3\right)
\end{aligned}
$$

Note that the $k's$ and $c's$ should be computed following the above order. For higher-order systems, the Runge-Kutta methods can be developed in a way that is entirely similar to second-order systems.

EXAMPLE 9.11

Solve the following initial-value problem using the Runge-Kutta method of order 4 with $N = 10$.

$$y''(t) - 3y' + 2y = 6e^{3t}, \quad 0 \le t \le 1$$

with initial conditions

$$y(0) = 1, \quad y'(0) = -1$$

Compare with the exact solution $y(t) = -8e^{2t} + 6e^{t} + 3e^{3t}$.

Using the substitution in (9.45), we get the reformulated problem:

$$
\begin{aligned}
x'(t) &= 3x - 2y + 6e^{3t}, & x(0) &= -1, \\
y'(t) &= x, & y(0) &= 1, \qquad 0 \le t \le 1.
\end{aligned}
$$

Thus,

$$f(t, x, y) = 3x - 2y + 6e^{3t} \quad \text{and} \quad g(t, x, y) = x.$$

m-function 9.6

The following MATLAB function **sys_rk4.m** solve the system 9.47 using the Runge-Kutta method of order 4. INPUTS are functions *f(t,x,y)* and *g(t,x,y)*; the initial and final values *a*, *b* of *t*; the initial conditions x_0, y_0 of *x* and *y*; the number of steps *n*. The input functions *f* and *g* should be defined as M-files.

```
function sys_rk4(f,g,a,b,x0,y0,n)
% Solve the system of differential equations x'=f(t,x,y), x(a)=x0 and
% y'=g(t,x,y), y(a)=y0 using the Runge-Kutta method of order 4.
fprintf('\n')
disp('      Solution of system of differential equations using RK4 ')
h=(b-a)/n;
x=x0;
y=y0;
disp('_____________________________________________________')
disp('  t      x      y      x(t)      y(t)   |x-x(t)|    |y-y(t)|  ')
disp('_____________________________________________________')
fprintf('\n')
fprintf('%8.2f    %10.3f %10.3f %10.3f %10.3f %8.2f %8.2f\n', a, x,
y, x, y, 0, 0)
for i=1:n
 % RK of order 4
 t=a+(i-1)*h;
 k1=feval(f,t,x,y);
 c1=feval(g,t,x,y);
 k2=feval(f,t+h/2,x+h*k1/2,y+h*c1/2);
 c2=feval(g,t+h/2,x+h*k1/2,y+h*c1/2);
 k3=feval(f,t+h/2,x+h*k2/2,y+h*c2/2);
 c3=feval(g,t+h/2,x+h*k2/2,y+h*c2/2);
 k4=feval(f,t+h,x+h*k3,y+h*c3);
 c4=feval(g,t+h,x+h*k3,y+h*c3);
 x=x+h*(k1+2*k2+2*k3+k4)/6;
 y=y+h*(c1+2*c2+2*c3+c4)/6;
 %If the exact solution is not known set xs='n' and ys='n'.
 % For a system with two Eqns. write the exact solutions
 % if known as xs=xs(t) and ys=ys(t).
 % For a second order IVP write the exact solution
 % if known as ys=ys(t) and set xs='n'.
 t=t+h;
 xs='n';
 ys='n';
 if (xs=='n' & ys=='n')
```

```
    fprintf('%8.2f    %10.3f %10.3f\n',t,x,y)
  elseif (ys~='n' & xs=='n')
    err2=abs(ys-y);
    fprintf('%8.2f    %10.3f %10.3f %10.3f %10.1e\n', t,x,y,ys,err2)
  else
    err1=abs(xs-x);
    err2=abs(ys-y);
    fprintf('%8.2f    %10.3f %10.3f %10.3f %10.3f %10.1e %10.1e
\n', t, x, y, xs, ys, err1, err2)
  end
end
```

We illustrate the use of the Runge-Kutta method of order 4 by estimating $y(0.1)$. With $h = 0.1$, we have

$$
\begin{aligned}
k_1 &= f(0, -1, 1) = 1 \\
c_1 &= g(0, -1, 1) = -1 \\
k_2 &= f(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(1), 1 + \frac{0.1}{2}(-1)) = 2.221 \\
c_2 &= g(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(1), 1 + \frac{0.1}{2}(-1)) = -0.95 \\
k_3 &= f(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(2.221), 1 + \frac{0.1}{2}(-0.95)) = 2.399155 \\
c_3 &= g(0 + \frac{0.1}{2}, -1 + \frac{0.1}{2}(2.221), 1 + \frac{0.1}{2}(-0.95)) = -0.88895 \\
k_4 &= f(0 + 0.1, -1 + (0.1)(2.399155), 1 + (0.1)(-0.88895)) = 3.99669 \\
c_4 &= g(0 + 0.1, -1 + (0.1)(2.399155), 1 + (0.1)(-0.88895)) = -0.76008
\end{aligned}
$$

Then,

$$
\begin{aligned}
x(0.1) &\approx -1 + \frac{0.1}{6}[1 + 2(2.221) + 2(2.399155) + 3.99669] \\
&\approx -0.762\,716\,4 \\
y(0.1) &\approx 1 + \frac{0.1}{6}[-1 + 2(-0.95) + 2(-0.88895) - 0.76008] \\
&\approx 0.909\,3669
\end{aligned}
$$

In Table 9.14, the numerical solution is continued and compared with values obtained from the analytical solution.

sysrk4('f1','g1',0,1,-1,1,10)

System of differential equation using Runge-Kutta method of order 4

t	x	y	x(t)	y(t)	|x-x(t)|	|y-y(t)|
0.0	-1.000	1.0000	-1.000	1.0000	0.000	0.000
0.1	-0.7627	0.9094		0.9094		1.29e-005
0.2	-0.1418	0.8601		0.8602		3.61e-005
0.3	1.0815	0.9009		0.9010		7.50e-005
0.4	3.2231	1.1068		1.1070		1.38e-004
0.5	6.7345	1.5909		1.5911		2.35e-004
0.6	12.2569	2.5203		2.5207		3.85e-004
0.7	20.6936	4.1388		4.1394		6.09e-004
0.8	33.3114	6.7976		6.7985		9.42e-004
0.9	51.8779	10.9982		10.9996		1.43e-003
1.0	78.8502	17.4517		17.4539		2.13e-003

Table 9.14 RK4 solution for Example 9.11.

EXAMPLE 9.12

Solve the following system of differential equations using the Runge-Kutta method of order 4 with $N = 10$.

$$\begin{aligned} x'(t) &= -3x + 4y, \quad x(0) = 1 \\ y'(t) &= -2x + 3y, \quad y(0) = 2, \quad 0 \le t \le 1. \end{aligned}$$

Compare the results with the values of the exact solution

$$x(t) = 3e^t - 2e^{-t} \quad \text{and} \quad y(t) = 3e^t - e^{-t}.$$

Set

$$f(t, x, y) = -3x + 4y \quad \text{and} \quad g(t, x, y) = -2x + 3y.$$

The numerical computations are given in Table 9.15 along with the true solution values for comparison.

```
» sysrk4('f2','g2',0,1,1,2,10)

System of differential equation using Runge-Kutta method of order 4
```

t	x	y	x(t)	y(t)	|x-x(t)|	|y-y(t)|
0.0	1	2	1	2	0	0
0.1	1.50584	2.41067	1.50584	2.41068	4.2e-007	3.4e-007
0.2	2.02675	2.84548	2.02675	2.84548	8.6e-007	7.1e-007
0.3	2.56794	3.30876	2.56794	3.30876	1.3e-006	1.1e-006
0.4	3.13483	3.80515	3.13483	3.80515	1.9e-006	1.6e-006
0.5	3.7331	4.33963	3.7331	4.33963	2.4e-006	2.2e-006
0.6	4.36873	4.91754	4.36873	4.91754	3.1e-006	2.8e-006
0.7	5.04808	5.54467	5.04809	5.54467	3.9e-006	3.6e-006
0.8	5.77796	6.22729	5.77796	6.22729	4.7e-006	4.4e-006
0.9	6.56566	6.97223	6.56567	6.97224	5.8e-006	5.4e-006
1.0	7.41908	7.78696	7.41909	7.78697	6.9e-006	6.6e-006

Table 9.15 RK4 solution for Example 9.12.

EXERCISE SET 9.10

1. Solve the following IVPs

   (a) $y'' = y' + ty^2,\ y(0) = 1,\ y'(0) = 0,\ 0 \le t \le 1$ with $N = 10$.

   (b) $y'' = 3y' - 2y, y(0) = 1,\ y'(0) = 1,\ 0 \le t \le 1$ with $N = 10$.

   (c) $y'' = ty' - 2y,\ y(0) = -1,\ y'(0) = 1,\ 0 \le t \le 1$ with $N = 10$.

   (d) $y'' = t^2y - ty + t,\ y(0) = 1,\ y'(0) = 1,\ 0 \le t \le 1$ with $N = 10$.

2. Solve the following systems of first-order differential equations

   (a) $\begin{aligned} x' &= t - y + x,\ x(0) = 1, \\ y' &= x + 2y, \quad y(0) = 0,\ 0 \le t \le 1, \text{ with } h = 0.2. \end{aligned}$

   (b) $\begin{aligned} x' &= 2 - 4y - x\sin t,\ x(0) = 0, \\ y' &= x, \qquad\qquad y(0) = 0,\ 0 \le t \le 1, \text{ with } h = 0.2. \end{aligned}$

   (c) $\begin{aligned} x' &= t^2 - 3x^2 + y,\ x(0) = 0, \\ y' &= 2t + x - y, \quad y(0) = 1,\ 0 \le t \le 1, \text{ with } h = 0.1. \end{aligned}$

(d) $\begin{array}{ll} x' = t^3 - y^2, & x(0) = 0, \\ y' = t + y - xy, & y(0) = 1,\ 0 \le t \le 1, \text{ with } h = 0.2. \end{array}$

9.11 IMPLICIT METHODS AND STIFF SYSTEMS

Consider the system of differential equations

$$\begin{aligned} x'(t) &= 131\,x - 931\,y, \qquad x(0) = 6 \\ y'(t) &= 133\,x - 933\,y, \qquad y(0) = -6 \end{aligned} \tag{9.49}$$

The exact solution of this system is

$$x(t) = 14e^{-2t} - 8e^{-800t}, \qquad y(t) = 2e^{-2t} - 8e^{-800t}.$$

Suppose we approximate the values of $x(0.1)$ and $y(0.1)$ using Euler's method with $h = 0.1$. The result is

$$x(0.1) \approx 643.2 \qquad \text{and} \qquad y(0.1) \approx 633.6$$

If we compare these approximate values to the exact ones $x(0.1) = 11.462$ and $y(0.1) = 1.638$, one can clearly see that the absolute error is large. What caused the error to be large is the presence of the two terms e^{-2t} and e^{-800t} in the exact solutions. Note that the exponents of these two terms are both negative and of different magnitude. In such cases, using an explicit method such as Euler's method with a small value for the step size leads to rapidly growing round-off error. Systems such (9.49) are called **stiff systems**.

The way to deal with stiff systems is to use an implicit method such as the Adams-Moulton method of order one known as the implicit Euler method given by

$$\begin{aligned} x_{i+1} &= x_i + hf(x_{i+1}, y_{i+1}) \\ y_{i+1} &= y_i + hf(x_{i+1}, y_{i+1}) \end{aligned} \tag{9.50}$$

A computer implementation of the implicit form of Euler's method produced the values in Table 9.16. The true solution values $\{x(t_k), y(t_k)\}$ are included for comparison. The results shown in Table 9.16 demonstrate the superiority of the implicit method.

t_k	$x(t_k)$	$y(t_k)$	x_k	y_k
0.1	11.462	1.6375	11.568	1.5679
0.01	13.72	1.9577	12.837	1.0719
0.001	10.377	−1.5986	9.5276	−2.4484
0.0001	6.6123	−5.3853	6.5898	−5.4078

Table 9.16 Comparison of the exact solution and implicit Euler's method for the system (9.49).

To understand better stiff systems, we consider the general form of a 2×2, linear system of differential equations. The matrix form of the system is

$$\begin{bmatrix} dx/dt \\ dy/dt \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{9.51}$$

It can be shown that the general solution of (9.51) is

$$\begin{bmatrix} x \\ y \end{bmatrix} = C_1 e^{\lambda_1 t}\mathbf{v}_1 + C_2 e^{\lambda_2 t}\mathbf{v}_2$$

where $\mathbf{v_1}$ and $\mathbf{v_2}$ are the eigenvectors of A and λ_1 and λ_2 are the corresponding eigenvalues. C_1 and C_2 are constants which depend on the initial conditions of the system.

A stiff system can be defined in terms of the eigenvalues of A. When the real parts of the eigenvalues of A are negative and differ largely in magnitude, we say that the system is stiff.

Here we used the implicit Euler method as an example to solve the system (9.49); however, its accuracy is limited to its order. To get a better accuracy, one can use the higher-order Adams-Moulton methods such as the the fourth-order Adams-Moulton method given by equation (9.37) in Section 9.8.

EXERCISE SET 9.11

1. Solve the following stiff IVP using Euler's method

(a)

$$x' = 21x - 211y, \quad x(0) = -94, \quad 0 \le t \le 1, \text{ with } h = 0.1$$
$$y' = 23x - 213y, \quad y(0) = 94, \quad 0 \le t \le 1, \text{ with } h = 0.1$$

Compare the approximate results with the exact solution $x(t) = 117e^{-190t} - 211e^{-2t}$ and $y(t) = -23e^{-2t} + 117e^{-190t}$.

(b)

$$x' = 3x - 37y, \quad x(0) = 16$$
$$y' = 5x - 39y, \quad y(0) = -16$$

Compare the approximate results with the exact solution $x(t) = 37e^{-2t} - 21e^{-34t}$ and $y(t) = 5e^{-2t} - 21e^{-34t}$.

9.12 PHASE PLANE ANALYSIS: CHAOTIC DIFFERENTIAL EQUATIONS

In this section, we outline a method for understanding the behavior of nonlinear systems of ordinary differential equations which is gaining in importance and popularity. We shall restrict ourselves to the case of two differential equations for the unknown functions $x = x(t)$, $y = y(t)$

$$\frac{dx}{dt} = f(x, y) \quad \text{and} \quad \frac{dy}{dt} = g(x, y). \tag{9.52}$$

The equations (9.52) represent a physical situation in which the time variable does not occur explicitly but only implicitly through the dependence of x and y on t and so describes a self-governing or autonomous system. We shall assume that f and g are smooth functions of x and y. Solutions to (9.52) subject to given initial conditions

$$x(t_0) = x_0 \quad \text{and} \quad y(t_0) = y_0 \tag{9.53}$$

are functions of time but here we wish to think of them as being the parametric equations of a curve

$$(x(t), \ y(t)) \tag{9.54}$$

in the xy-plane passing through the given point (x_0, y_0). This curve is called the orbit on trajectory of the solution and the xy-plane is called the **phase plane**.

EXAMPLE 9.13

The solution to

$$\frac{dx}{dt} = -(x+y) \quad \text{and} \quad \frac{dy}{dt} = x - y$$

passing through $(1, 0)$ is $x = e^{-t}\cos t$ and $y = e^{-t}\sin t$. The trajectory is depicted in Figure 9.4.

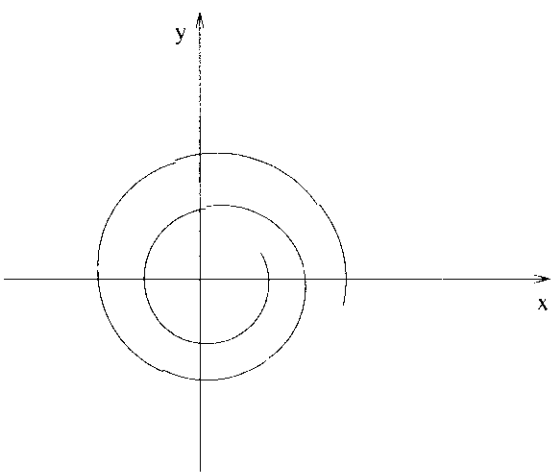


FIGURE 9.4
Spiral into a fixed point.

Observe that in this picture, the trajectory spirals into the origin $(0, 0)$ which is an equilibrium or fixed point for the system (9.52). The equilibrium or fixed points are points in the phase plane where the system remains unchanged so that $dx/dt = 0$ and $dy/dt = 0$ at such a point.

More precisely, (x^*, y^*) is an equilibrium or a fixed point for (9.52) if

$$f(x^*, y^*) = 0 \quad \text{and} \quad g(x^*, y^*) = 0. \tag{9.55}$$

EXAMPLE 9.14
Find the fixed points for

$$\frac{dx}{dt} = y + 3x^2 \quad \text{and} \quad \frac{dy}{dt} = x - 3y^2.$$

The fixed points are the solutions to the system

$$\begin{cases} y + 3x^2 = 0 \\ x - 3y^2 = 0 \end{cases}.$$

By inspection one sees that $x = 0$ if and only if $y = 0$ so $(0, 0)$ is a fixed point. If $x \neq 0$, then $y \neq 0$ as well and the equations show that $x > 0$ and $y < 0$ in that case. We find that $\left(\frac{1}{3}, -\frac{1}{3}\right)$ is the only other fixed point.

Nonlinear equations are often "linearized" in the neighborhood of fixed points, and very often the behavior of the linear system is very close to that of the nonlinear system. The general theory would take us too far a field to describe here, but in the problems we shall explore the behavior of solutions to nonlinear problems on the basis of concrete examples.

To guide us in our investigation, let us make the following observations. Suppose (x^*, y^*) is a fixed point for (9.52). Then near (x^*, y^*)

$$\begin{aligned} f(x, y) &= f_x(x^*, y^*)(x - x^*) + f_y(x^*, y^*)(y - y^*) + \cdots \\ g(x, y) &= g_x(x^*, y^*)(x - x^*) + g_y(x^*, y^*)(y - y^*) + \cdots \end{aligned}$$

Let $u = x - x^*$ and $v = y - y^*$ so that sufficiently close to (x^*, y^*), the solutions $(u(t), v(t))$ to the linearized system

$$\begin{aligned} \frac{du}{dt} &= f_x(x^*, y^*)\, u + f_y(x^*, y^*)\, v \\ \frac{dv}{dt} &= g_x(x^*, y^*)\, u + g_y(x^*, y^*)\, v \end{aligned} \tag{9.56}$$

should furnish a good approximation to $(u(t), v(t))$, that is, we would expect

$$x(t) \approx x^* + u(t) \quad \text{and} \quad y(t) \approx y^* + v(t).$$

EXAMPLE 9.15
Consider the linearization of the system in Example 9.14 about the fixed points $(0,0)$ and $\left(\frac{1}{3}, -\frac{1}{3}\right)$.

The linearization of the system about the fixed points $(0,0)$ is

$$\frac{du}{dt} = v \quad \text{and} \quad \frac{dv}{dt} = u.$$

The linearization of the system about the point $\left(\frac{1}{3}, -\frac{1}{3}\right)$ is

$$\frac{du}{dt} = 2u + v \quad \text{and} \quad \frac{dv}{dt} = u + 2v.$$

The behavior of the solutions to the linearized problem depends on the eigenvalues of the Jacobian matrix

$$\begin{pmatrix} f_x(x^*, y^*) & f_y(x^*, y^*) \\ g_x(x^*, y^*) & g_y(x^*, y^*) \end{pmatrix} \tag{9.57}$$

If the real parts of the eigenvalues are nonzero, then, at least qualitatively, the solutions to the linearized problems will simulate those of the nonlinear problems quite well, whereas if the real parts are zero, that may not be the case.

It is also possible for solutions to spiral into or away from simple closed curves which are, themselves, solutions. These curves are called limit cycles. In general, finding limit cycles can be difficult analytically, but experimentally with the help of a computer they can readily be found.

EXAMPLE 9.16
Consider the system

$$\frac{dx}{dt} = -y + x\left(1 - x^2 - y^2\right) \quad \text{and} \quad \frac{dy}{dt} = x + y\left(1 - x^2 - y^2\right).$$

This system has one fixed point, $(0,0)$. However, if one sets $x = \gamma\cos\theta$ and $y = \gamma\sin\theta$ and rewrites the equation in terms of polar

coordinates, one finds

$$\frac{d\gamma}{dt} = \gamma\left(1-\gamma^2\right) \quad \text{and} \quad \frac{d\theta}{dt} = 1.$$

We see that the circle $\gamma = 1$ and $\theta = t + \theta_0$ is a circle which also solves the differential equations. In the problems, we will verify numerically that this circle is also a limit cycle.

We shall be content with this informal discussion of the general properties of nonlinear systems. When the systems are forced with a time dependent forcing function, the solutions can behave erratically and lead naturally to a discussion of chaotic, dynamical systems. The fixed points themselves can be further classified and studied, but that will not be done here.

Remark: Differential equations whose independent variable is the time t are called dynamical system. The long-term behavior of solutions to dynamical systems can be complicated and unpredictable. The modern theory includes the topics of stability, asymptotic stability, periodicity, limit cycles, and chaos.

EXERCISE SET 9.12

Recall that every second order differential equation

$$\frac{d^2x}{dt} = F\left(t, x, \frac{dx}{dt}\right)$$

can be reduced to a system of two first order equations by letting $dx/dt = y$. The resulting system is

$$\frac{dx}{dt} = y \quad \text{and} \quad \frac{dy}{dt} = F(t, x, y).$$

Rewrite the differential equations in Problems 1–4 as equivalent systems.

1. $\frac{d^2x}{dt^2} = -\frac{dx}{dt} + \sin x$
2. $\frac{d^2x}{dt^2} = (1 - x^2)\frac{dx}{dt} + x - x^3$
3. $\frac{d^2x}{dt^2} = \sin\left(x + \frac{dx}{dt}\right)$

4. $\frac{d^2x}{dt^2} = -t\frac{dx}{dt} + x^2 \cos t$

In Problems 5–8, find the fixed points. The fixed points for a second order differential equation are the points in the phase plane for the equivalent second order system and hence, all lie on the x-axis.

5. $\frac{d^2x}{dt^2} = (1 - x^2)\frac{dx}{dt} + (x - x^3)$

6. $\frac{dx}{dt} = \sin(x - y), \quad \frac{dy}{dx} = y \cos x$

7. $\frac{dx}{dt} = 1 - e^{-x} - y, \quad \frac{dy}{dx} = \sin y + x(1 - x)$

8. $\frac{dx}{dt} = x - xy, \quad \frac{dy}{dt} = -y + yx^3$

The following problems deal with constructing phase plane plots and comparing the plots of the solutions of the nonlinear problems with those of the associated linear problems.

9. Consider the system of Example 9.16:

   (a) Show analytically that $x = \cos t$, $y = \sin t$ is a solution to the differential equation which satisfies the initial condition $x(0) = 1$, $y(0) = 0$.

   (b) Show that near the fixed point $(0, 0)$, the linearized problem is
   $$\frac{du}{dt} = u - v, \quad \frac{dv}{dt} = u + v$$
   Solve this system and draw the trajectories for the sets of initial conditions: $(x(0) = 0.1, y(0) = 0)$; $(x(0) = 0.5, y(0) = 0)$; $(x(0) = 2, y(0) = 0)$; and $(x(0) = 5, y(0) = 0)$.

   (c) Solve the nonlinear system using the same initial conditions as in (b) and compare the two sets of phase plots.

10. Consider the two differential equations:

    (a) $\frac{d^2x}{dt^2} + \left(\frac{dx}{dt}\right)^3 + x = 0$

    (b) $\left(\frac{d^2x}{dt^2}\right) - \left(\frac{dx}{dt}\right)^3 + x = 0$

    In both cases, the origin is the unique fixed point and the linear form of both equations is $\frac{d^2u}{dt^2} + u = 0$. Solve both the nonlinear equations subject to the initial conditions $x(0) = 0$, $x'(0) = 1$, and compare the phase plots so obtained with each other and with that obtained for the linearized problem.

11. Does the equation $\frac{d^2x}{dt^2} + (x^2 - 1)\frac{dx}{dt} + x = 0$ have a limit cycle? Try various initial conditions and carry out some numerical experiments.

12. Consider the following differential equations:

    (a) $\frac{d^2x}{dt^2} + x + x^3 = 0$
    (b) $\frac{d^2x}{dt^2} - x + x^3 = 0$
    (c) $\frac{d^2x}{dt^2} + x - x^3 = 0$

    Make phase plane plots for each of them for each of initial conditions: $(x(0) = .5, x'(0) = 0)$; $(x(0) = 1.5, x'(0) = 0)$; and $(x(0) = 4, x'(0) = 0)$.

13. Forcing introduces additional complications. Make phase plane plots for the solutions to the following differential equations:

    (a) $\frac{d^2x}{dt^2} + x + x^3 = \cos t$
    (b) $\frac{d^2x}{dt^2} + x - x^3 = \cos t$

    subject to the initial conditions $x(0) = 0$ and $x'(0) = 0$.

14. Make phase plane plots for each of the following differential equations:

    (a) $\frac{d^2x}{dt^2} + \frac{dx}{dt} + x = \cos t$
    (b) $\frac{d^2x}{dt^2} - \frac{dx}{dt} + x = \cos t$
    (c) $\frac{d^2x}{dt^2} + \frac{dx}{dt} - \left(\frac{dx}{dt}\right)^3 + x = \cos t$
    (d) $\frac{d^2x}{dt^2} + \frac{dx}{dt} + \left(\frac{dx}{dt}\right)^3 + x = \cos t$
    (e) $\frac{d^2x}{dt^2} - \frac{dx}{dt} + \left(\frac{dx}{dt}\right)^3 + x = \cos t$

    subject to the initial conditions $x(0) = 0$ and $x'(0) = 0$.

APPLIED PROBLEMS FOR CHAPTER 9

1. The differential equation

$$R\frac{dQ}{dt} + \frac{Q}{C} = V$$

describes the charge Q on a capacitor with capacitance C during a charging process involving a resistance R and an electromotive force V. If the charge is 0 when $t = 0$, approximate Q in the time interval $[0, 4]$ with $R = 2$, $C = 3$, and $V = 4$ using the Runge-Kutta of order 4.

2. Suppose an undamped weight of mass m is attached to a spring having spring constant k, and external periodic force given by $F = \sin \alpha t$ is applied to the weight. The differential equation that describes the motion of the weight y is

$$\frac{d^2y}{dt^2} + \omega^2 y = \frac{F}{m} \sin \alpha t$$

where $\omega^2 = k/m$. If $k = 2.7 \times 10^2 N/m$, $m = 0.55kg$, $F = 2N$, and $\alpha = 10 \times s^{-1}$, approximate y at $t = 3\ sec$.

3. An automobile heading light mirror is designed to reflect the light given off by the headlamp in rays parallel to the read surface. By using the principle of optics that the angle of light incidence equals the angle of light reflection, we can derive a first-order differential equation that models the desired shape of the mirror

$$\frac{dy}{dx} = \frac{-x + (x^2 + y^2)^{1/2}}{y}$$

The mirror is designed so that the distance of the mirror directly above the lamp is 1 cm, so $y(0) = 1$. Use the Runge-Kutta method of order 4 with $h = 0.1$ to estimate y at $x = 1$ and $x = 2$.

4. Suppose a 24-lb weight stretches a spring 1 ft beyond its natural length. If the weight is pushed downward from its equilibrium position with an initial velocity of 2 ft/sec and if the damping force is $-9(dy/dt)$, the displacement y of the weight at any time t is given by the differential equation

$$\frac{d^2y}{dt^2} + 12\frac{dy}{dt} + 32y = 0$$

with $y(0) = 0$ and $y'(0) = 2$. Compute the displacement of the spring at $t = 4$ using the Runge-Kutta method of order 4 for system of equations.

5. The opening and closing of small, high-performance hydraulic valves used to control guidance jets in spacecrafts are accomplished by the use of a torque motor. If the mass m of the rotor is 32 slugs, the coefficient of viscous damping is 28 lb-sec/ft, the spring constant is 96 lb/ft, the radius r is 1 ft, and the motor is driven by a torque M equal to 32 $\cos t$ (in lb/ft), then the IVP describing the rotation of the motor shaft is

$$\theta'' + 4\theta' + 3\theta = \cos t, \quad \theta(0) = \theta'(0) = 0$$

Approximate the angular displacement of the motor shaft at $t = 4$ using the Runge-Kutta method of order 4 for system of equations.

6. An object of mass m is released from a hot-air balloon. The velocity of the object after t seconds is given by

$$\frac{dv}{dt} + cv = g \quad \text{with } v(0) = 0$$

Given that $c = 2.2$ and $g = 32$, use Euler's method to approximate the velocity of the object after 3 sec.

7. The population in a city changes at the rate

$$\frac{dp}{dt} = (k_0 + k_1 y)y$$

where $p(t)$ denotes the number of members in the population at time t (in years) and k_0, k_1 are specified constants. Given that the initial population size is 1000 and $k_0 = 0.02$, $k_1 = 0.4$, approximate the size of the population after 10 years, using the Runge-Kutta method of order 4 with $N = 10$.

Chapter 10

Boundary-Value Problems

All the problems discussed in Chapter 9 required conditions on $y(t)$ and its derivatives at only one point which define the initial value. For that reason, such problems are termed initial-value problems. However, in many problems, conditions are specified at more than one point. Such problems are classified as **boundary-value problems** (BVP). These problems arise in the study of beam deflection, heat flow, and various physical problems. In this chapter we shall develop two principal classes of numerical methods used for solving the following boundary-value problems:

The linear second-order boundary-value problem,

$$\begin{cases} y''(x) + p(x)y' + q(x)y = r(x) \\ y(a) = \alpha, \quad y(b) = \beta \end{cases} \tag{10.1}$$

and the nonlinear second-order boundary-value problem,

$$\begin{cases} y''(x) = f(x, y, y') \\ y(a) = \alpha, \quad y(b) = \beta. \end{cases} \tag{10.2}$$

10.1 FINITE-DIFFERENCE METHODS

In this section, we assume that we have a linear differential equation. The problem we want to solve is

$$y''(x) + p(x)y' + q(x)y = r(x), \quad a \le x \le b \tag{10.3}$$

subject to the boundary conditions

$$y(a) = \alpha \quad \text{and} \quad y(b) = \beta. \tag{10.4}$$

To accomplish this, let us first divide the interval $[a, b]$ into N equal subintervals, each of size h, with

$$h = \frac{b-a}{N} \quad \text{and} \quad x_i = a + hi, \quad i = 0, 1, \ldots, N.$$

The **finite-difference** method consists of replacing every derivative in Eqn. (10.3) by finite difference approximations such as the central divided difference approximations

$$\begin{aligned} y'(x_i) &\approx \frac{1}{2h}\left[y(x_{i+1}) - y(x_{i-1})\right] \\ y''(x_i) &\approx \frac{1}{h^2}\left[y(x_{i+1}) - 2y(x_i) + y(x_{i-1})\right]. \end{aligned} \tag{10.5}$$

Observe that, using central divided difference approximations to derivatives will result in a finite-difference method with a truncation error of order $O(h^2)$.

Substituting (10.5) into Eqn. (10.3), and rearranging, we get the difference equation

$$\left[1 - \frac{h}{2}p_i\right] y_{i-1} + (-2 + h^2 q_i) y_i + \left[1 + \frac{h}{2}p_i\right] y_{i+1} = h^2 r_i \tag{10.6}$$

$i = 1, 2, \ldots, N-1$
and

$$y_0 = \alpha \quad \text{and} \quad y_N = \beta \tag{10.7}$$

where $y_i \approx y(x_i)$, $p_i = p(x_i)$, $q_i = q(x_i)$, and $r_i = r(x_i)$.
The equations given by (10.6) form a linear system in $(N-1)$ equations and $(N-1)$ unknowns. It can be represented in the tridiagonal matrix form

$$A\mathbf{y} = \mathbf{b}, \tag{10.8}$$

or

$$\begin{bmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & 0 & \\ & a_3 & b_3 & c_3 & & \\ & & & \ddots & & \\ & 0 & & a_{N-2} & b_{N-2} & c_{N-2} \\ & & & & a_{N-1} & b_{N-1} \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{N-2} \\ y_{N-1} \end{bmatrix} = \begin{bmatrix} d_1 - a_1\alpha \\ d_2 \\ d_3 \\ \vdots \\ d_{N-2} \\ d_{N-1} - c_{N-1}\beta \end{bmatrix}$$

where a_i, b_i, c_i, and d_i are defined by

$$a_i = 1 - \frac{h}{2}p_i, \quad b_i = -2 + h^2 q_i,$$
$$c_i = 1 + \frac{h}{2}p_i, \quad d_i = h^2 r_i \tag{10.9}$$

for $i = 1, \ldots, N-1$.

So, our problem now reduces to solving the tridiagonal system (10.8) whose solution will give approximate values for the solution of the boundary-value problem at discrete points on the interval $[a, b]$.

To obtain a better accuracy for Eqn. (10.3), we could use more accurate finite-difference approximations to the derivatives. But, the disadvantage of doing this is that the resulting system of equations would not be tridiagonal and more calculations would be required. Another way of insuring accuracy is to solve the linear system for smaller values of h, and compare the solutions at the same mesh points; the round-off error, however, will eventually increase and may become large.

EXAMPLE 10.1

Solve the following boundary-value problem

$$\begin{cases} y'' + (x+1)y' - 2y = (1-x^2)e^{-x}, & 0 \le x \le 1 \\ y(0) = -1, \quad y(1) = 0 \end{cases}$$

using the above finite difference method with $h = 0.2$. Compare the results with the exact solution $y = (x-1)e^{-x}$.

In this example: $p(x) = x + 1$, $q(x) = -2$, and $r(x) = (1 - x^2)e^{-x}$; hence, Eqn. (10.6) yields

$$[1 - 0.1(x_i + 1)]y_{i-1} + (-2 - 0.08)y_i + [1 + 0.1(x_i + 1)]y_{i+1} = \\ +0.04(1 - x_i^2)e^{-x_i}$$

and

$$y_0 = -1 \quad \text{and} \quad y_5 = 0$$

where $x_i = 0.2i$, $i = 1, 2, 3, 4$.

The matrix formulation is

$$\begin{bmatrix} -2.08 & 1.12 & & \\ 0.86 & -2.08 & 1.14 & \\ & 0.84 & 2.08 & 1.16 \\ & & 0.82 & -2.08 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 0.91143926 \\ 0.02252275 \\ 0.01404958 \\ 0.00647034 \end{bmatrix}$$

The solution of this set is given in Table 10.1. For comparison, the table also gives values calculated from the analytical solution.

x	F. Diff. Method	Exact Sol.	Error
0.0	-1.00000	-1.00000	0.000000
0.2	-0.65413	-0.65498	0.000854
0.4	-0.40103	-0.40219	0.001163
0.6	-0.21848	-0.21952	0.001047
0.8	-0.08924	-0.08987	0.000624
1.0	0.00000	0.00000	0.000000

Table 10.1 The finite difference method for Example 10.1.

EXAMPLE 10.2

Solve the following boundary-value problem

$$y'' + e^x y' - xy = e^{-x}(-x^2 + 2x - 3) - x + 2$$

$$y(0) = -1 \quad \text{and} \quad y(1) = 0$$

using the finite difference method described in Section 10.1 with $h = 0.1$. Compare the results with the exact solution $y = (x - 1)e^{-x}$.

Set

$$p(x) = e^x, \quad q(x) = -x \quad \text{and} \quad r(x) = e^{-x}(-x^2 + 2x - 3) - x + 2$$

to get the results in Table 10.2.

m-function 10.1

The following MATLAB function **finitediff.m** finds the solution of the linear BVP (10.1) using the finite difference method described in sec. 10.1. INPUTS are functions *p(x), q(x), r(x)*; the end points *aa*, *bb*; the boundary conditions y_0 and y_n; the number of steps *n*. The input functions *p, q,* and *r* should be defined as M-files.

```
function finitediff(p,q,r,aa,bb,y0,yn,n)
% Solve the second order linear BVP using a finite difference
% method.
fprintf('\n')
h=(bb-aa)/n;
blank='     ';
disp(['Finite difference method with h=',num2str(h)])
fprintf('\n')
for i=1:n-1
  x=aa+i*h;
  if (i~=1)
    a(i-1)=1-h/2*feval(p,x); %Compute the subdiagonal of A.
  end
  b(i)=-2+h^2*feval(q,x);  %Compute the main diagonal of A.
  if (i~=n-1)
    c(i)=1+h/2*feval(p,x); %Compute the superdiagonal of A.
  end
end
disp(' The subdiagonal of A =')
disp(a)
disp(' The main diagonal of A =')
disp(b)
disp(' The superdiagonal diagonal of A =')
disp(c)
% Compute the coefficient vector B.
d(1)=h^2*feval(r,aa+h)-y0*(1-h/2*feval(p,aa+h));
d(n-1)=h^2*feval(r,bb-h)-yn*(1+h/2*feval(p,bb-h));
for i=2:n-2
  x=aa+i*h;
  d(i)=h^2*feval(r,x);
end
fprintf('\n')
disp(' The Coefficient vector B" =')
disp(d)
disp(' The solution of the BVP =')
fprintf('\n')
```

```
disp('   xi                yi                y(xi)              error')
fprintf('\n')
% Solve the tridiagonal system Ax=B.
for i=2:n-1
  ymult=a(i-1)/b(i-1);
  b(i)=b(i)-ymult*c(i-1);
  d(i)=d(i)-ymult*d(i-1);
end
y(n)=yn;
y(n-1)=d(n-1)/b(n-1);
for i=n-2:-1:1
  y(i)=(d(i)-c(i)*y(i+1))/b(i);
end
fprintf('%6.2f  %12.6f  %12.6f  %10.2e\n',aa,y0,y0,0)
for i=1:n
  x=aa+i*h;
%Write the exact solution if known as s=s(x) otherwise set s='n'.
  s='n';
  if (s=='n')
    fprintf('%6.2f %12.6f\n',x,y(i))
  else
    err=abs(s-y(i));
    fprintf('%6.2f  %12.6f  %12.6f  %10.2e\n',x,y(i),s,err)
  end
end
```

EXERCISE SET 10.1

1. Show that $y(x) = x + 1/x$ is a solution of the boundary-value problem

$$y'' = 2y^3 - 6y - 2x^3, \quad y(1) = 2, \; y(2) = 5/2.$$

2. Solve the following boundary-value problems using the finite difference method developed in Section 10.1.

   (a) $y'' + 9y = 0, \; y(0) = 1, \; y(\pi) = -1, \quad 0 \le x \le \pi, \quad h = \pi/6.$

   (b) $y'' + y = x, \; y(0) = y(\pi) = 0, \quad 0 \le x \le \pi, \quad h = \pi/4.$

   (c) $y'' + 2y = \cos x, \; y(0) = y(1) = 0, \quad 0 \le x \le 1, \quad h = 0.1.$

   (d) $y'' + y' + xy = 0, \; y(0) = 1, \; y(1) = 0, \quad 0 \le x \le 1, \quad h = 0.2.$

```
» finited( 'p', 'q', 'r', 0, 1, -1, 0, 10)

    Finite difference method with h = 0.1

The subdiagonal of A =

  0.9389   0.9325   0.9254   0.9176   0.9089   0.8993   0.8887   0.8770

The main diagonal of A =

 -2.0010  -2.0020  -2.0030  -2.0040  -2.0050  -2.0060  -2.0070  -2.0080  -2.0090

The superdiagonal diagonal of A =

  1.0553   1.0611   1.0675   1.0746   1.0824   1.0911   1.1007   1.1113

The Coefficient vector B' =

  0.9383  -0.0036  -0.0014   0.0002   0.0014   0.0021   0.0026   0.0028   0.0028

The solution of the BVP =

      xi            yi            y(xi)          error
-----------------------------------------------------------------------------
      0             -1            -1             0.0E+00
      0.1           -0.814197     -0.814354      1.6E-04
      0.2           -0.654714     -0.654985      2.7E-04
      0.3           -0.518229     -0.518573      3.4E-04
      0.4           -0.401815     -0.402192      3.8E-04
      0.5           -0.30289      -0.303265      3.8E-04
      0.6           -0.219182     -0.219525      3.4E-04
      0.7           -0.148691     -0.148976      2.8E-04
      0.8           -0.089661     -0.089866      2.0E-04
      0.9           -0.040549     -0.040657      1.1E-04
      1             0             0              0.0E+00
```

Table 10.2 The finite difference method for Example 10.2.

3. Apply a finite difference method to solve the boundary-value problem
$y'' + x^2y' - 4xy = 0,\ y(0) = 0,\ y(1) = 5, \quad 0 \le x \le 1$
with $h = 0.1$. Compare with the exact solution $y = x^4 + 4x$.

10.2 SHOOTING METHODS

10.2.1 The nonlinear case

The finite difference method developed in the previous section works reasonably well for linear boundary-value problems and does not present problems of instability. For a boundary-value problem involving a nonlinear differential equation, these methods run into problems, in that, the resulting system is nonlinear. In such situations, it is often preferable to use the so-called **shooting** method, which we now discuss.

Consider the boundary-value problem

$$y''(x) = f(x, y, y'), \quad a \le x \le b \tag{10.10}$$

with the boundary conditions

$$y(a) = \alpha, \quad y(b) = \beta.$$

The shooting method consists of converting Problem (10.10) to a second-order initial-value problem of the form

$$y''(x) = f(x, y, y'), \quad a \le x \le b \tag{10.11}$$

subject to the initial conditions

$$y(a) = \alpha, \quad \text{and} \quad y'(a) = \gamma.$$

Since the condition $y'(a) = \gamma$ is not known, we arbitrarily choose a value γ_1, and the integration from a to b of the initial-value problem (10.11) is carried out by using any standard method for initial-value problems. Let $y(\gamma_1; b)$ be the computed solution at $x = b$. The problem is to determine γ, so that

$$y(\gamma; b) - \beta = 0. \tag{10.12}$$

Again, we arbitrarily choose another value, γ_2, and the procedure is repeated. Let $y(\gamma_2; b)$ be the computed solution at $x = b$ (see Figure 10.1). Having obtained two values of y at $x = b$, the next trial value, γ_3, for the initial solution can now be generated by using linear interpolation

$$\gamma_3 = \gamma_1 + (\gamma_2 - \gamma_1) \frac{\beta - y(\gamma_1; b)}{y(\gamma_2; b) - y(\gamma_1; b)}. \tag{10.13}$$

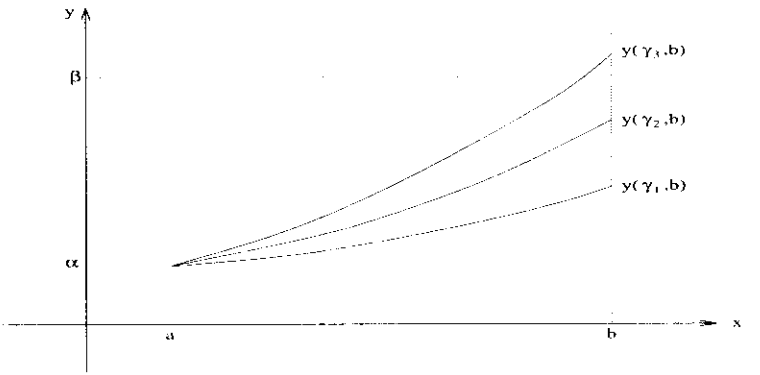


FIGURE 10.1
The shooting method and the first three approximations to its solution.

The general form of Eqn. (10.13) is

$$\gamma_{k+1} = \gamma_{k-1} + (\gamma_k - \gamma_{k-1})\frac{\beta - y(\gamma_{k-1}; b)}{y(\gamma_k; b) - y(\gamma_{k-1}; b)}, \quad k = 2, 3, \ldots \quad (10.14)$$

We now repeat the procedure by using the value γ_3 to get $y(\gamma_3; b)$. This procedure is repeated until convergence has been obtained, i.e.,

$$|y(\gamma_k; b) - \beta| < \epsilon \quad \text{for a specified tolerance } \epsilon. \quad (10.15)$$

The method we have illustrated is known as the shooting method. One should note that in order for the method to converge, a good choice of γ_1 and γ_2 is necessary.

There are ways of improving the procedure (10.14) by using higher-order interpolation to generate the sequence of values γ_4, γ_5, $\gamma_6, \ldots$. For example, having obtained $y(\gamma_3; b)$, we can utilize the three values $y(\gamma_1; b)$, $y(\gamma_2; b)$, and $y(\gamma_3; b)$ to generate γ_4 using a quadratic interpolation. A disadvantage, of course, of doing this is that more calculations will be needed.

EXAMPLE 10.3

Use the nonlinear shooting method with $N = 20$ to solve the following boundary-value problem with an accuracy of 10^{-6}.

$$yy'' = -(y')^2, \quad 1 \le x \le 3$$

with

$$y(1) = \sqrt{2} \quad \text{and} \quad y(3) = 2.$$

Compare your results with the exact solution $y = \sqrt{1 + x}$.

For this example we have $f(x, y, y') = -\frac{(y')^2}{y}$, $a = 1$ and $b = 3$. As a first approximation, we set $\gamma_1 = y(1) \approx 1.4$, and we replace the boundary-value problem with the initial-value problem

$$\begin{aligned} yy'' &= -(y')^2, \quad 1 \le t \le 3 \\ y(1) &= \sqrt{2} \quad \text{and} \quad y'(1) = \gamma_1 = 1.4 \end{aligned}$$

Using the fourth-order Runge-Kutta method with $n = 20$, we find that

$$y(\gamma_1, 3) = 3.14953128 \quad \text{and} \quad |y(\gamma_1, 3) - 2| = 1.14953128.$$

We repeat the procedure using a different estimate of $y'(1)$. For the second approximation, we set $\gamma_2 = \frac{2-\sqrt{2}}{3-1} \approx 0.3$; this leads to

$$y(\gamma_2, 3) = 1.92277306 \quad \text{and} \quad |y(\gamma_2, 3) - 2| = 0.07722694.$$

For the third approximation to γ_3, we use equation (10.13) to get

$$\gamma_3 = 1.4 + (0.3 - 1.4)\frac{2 - 3.14953128}{1.92277306 - 3.14953128} \approx 0.369\,247\,25$$

which leads to

$$y(\gamma_3, 3) = 2.02207265 \quad \text{and} \quad |y(\gamma_2, 3) - 2| = 0.02207265.$$

The results of further calculations of the $y(\gamma_k, 3)$ is given in Table 10.3. After six attempts, a value $\gamma_6 = 0.35355340$ has been located for which $y(\gamma_6, 3) = 2$ to within 10 decimal places.

ti	yi	y(ti)	Error
1.0	1.41421356	1.41421356	0
1.1	1.44913767	1.44913767	1.41E-09
1.2	1.48323970	1.48323970	2.39E-09
1.3	1.51657509	1.51657509	3.05E-09
1.4	1.54919334	1.54919334	3.48E-09
1.5	1.58113883	1.58113883	3.71E-09
1.6	1.61245155	1.61245155	3.81E-09
1.7	1.64316767	1.64316767	3.78E-09
1.8	1.67332005	1.67332005	3.68E-09
1.9	1.70293863	1.70293864	3.50E-09
2.0	1.73205080	1.73205081	3.27E-09
2.1	1.76068168	1.76068169	3.00E-09
2.2	1.78885438	1.78885438	2.70E-09
2.3	1.81659021	1.81659021	2.37E-09
2.4	1.84390889	1.84390889	2.03E-09
2.5	1.87082869	1.87082869	1.67E-09
2.6	1.89736659	1.89736660	1.29E-09
2.7	1.92353841	1.92353841	9.14E-10
2.8	1.94935887	1.94935887	5.29E-10
2.9	1.97484177	1.97484177	1.40E-10
3.0	2	2	2.50E-10

Table 10.3 The nonlinear shooting method for Example 10.3.

10.2.2 The linear case

Consider the linear second-order boundary-value problem,

$$\begin{cases} y'' + p(x)\,y' + q(x)\,y = r(x) \\ \quad y(a) = \alpha, \quad y(b) = \beta. \end{cases} \tag{10.16}$$

The solution to this problem can be obtained by forming a linear combination of the solution to the following two second-order initial-value problems

$$u'' + p(x)u' + q(x)u = r(x), \quad u(a) = \alpha, \quad u'(a) = 0 \tag{10.17}$$

and

$$v'' + p(x)v' + q(x)v = 0, \quad v(a) = 0, \quad v'(a) = 1. \tag{10.18}$$

We claim that the linear combination

$$y(x) = u(x) + Av(x) \tag{10.19}$$

is a solution to the differential equation $y'' + p(x)y' + q(x)y = r(x)$ in (10.16).

To show that, we substitute Eqn. (10.19) into the differential equation in (10.16) to get

$$u''(x) + Av''(x) + p(x)[u'(x) + Av'(x)] + q(x)[u(x) + Av(x)] = r(x)$$
$$\underbrace{u''(x) + p(x)u'(x) + q(x)u(x)}_{r(x)} + A\underbrace{[v''(x) + p(x)v'(x) + q(x)v(x)]}_{0} = r(x).$$

To find the constant A, we use the boundary conditions in (10.16) to get

$$y(a) = u(a) + Av(a) = \alpha + 0 = \alpha$$
$$y(b) = u(b) + Av(b) = \beta$$

from which

$$A = \frac{\beta - u(b)}{v(b)}. \tag{10.20}$$

Therefore, if $v(b) \neq 0$, the solution to boundary-value problem (10.16) is

$$y(x) = u(x) + \frac{\beta - u(b)}{v(b)} v(x). \tag{10.21}$$

The solutions to the second-order initial-value problems (10.17) and (10.18) was described in Section 9.10 and can be obtained by using the fourth-order Runge-Kutta method.

EXAMPLE 10.4

Solve the boundary-value problem

$$y'' + (x + 1)y' - 2y = (1 - x^2)e^{-x}, \quad 0 \le x \le 1, \quad y(0) = y(1) = 0$$

using the linear shooting method with $h = 0.2$.

The functions p, q, and r are $p(x) = x + 1$, $q(x) = -2$, and $r(x) = (1 - x^2)e^{-x}$, respectively. The numerical solution to this problem is

$$y(x) = u(x) - \frac{u(1)}{v(1)} v(x) \tag{10.22}$$

where $u(x)$ and $v(x)$ are the solutions of the initial-value problems

$$\begin{aligned} &u'' + (x+1)u' - 2u = (1-x^2)\,e^{-x}, \quad u(0) = 0, \quad u'(0) = 0, \\ &v'' + (x+1)v' - 2v = 0, \qquad\qquad\qquad\quad v(0) = 0, \quad v'(0) = 1, \end{aligned}$$

respectively. They can be expressed in the form

$$\begin{aligned} &u' = w, \qquad\qquad\qquad\qquad\qquad u(0) = 0, \\ &w' = 2u - (x+1)w + (1-x^2)\,e^{-x}, \; w(0) = 0, \quad 0 \le x \le 1 \end{aligned} \tag{10.23}$$

and

$$\begin{aligned} &v' = z, \qquad\qquad\qquad v(0) = 0, \\ &z' = 2v - (x+1)\,z, \; z(0) = 1, \quad 0 \le x \le 1 \end{aligned} \tag{10.24}$$

The Runge-Kutta method of order 4 is used to construct numerical solutions u_i and v_i to the linear systems (10.23) and (10.24). The approximations of u_i and v_i are given in Table 10.4.

x	ui	vi
0	0	0
0.2	0.01743428	0.18251267
0.4	0.06017154	0.33919207
0.6	0.11620204	0.48140428
0.8	0.17679396	0.61776325
1	0.23633393	0.75454368

Table 10.4 Numerical solutions of systems (10.23) and (10.24).

For example, using (10.22) and the Table 10.4 the approximate solution y_2 at $x = 0.4$ is

$$\begin{aligned} y_2 &= u(0.4) - \frac{u(1)}{v(1)} v(0.4) \\ &= 0.06017154 - \frac{0.23633393}{0.75454368}(0.33919207) \\ &= -0.046068294. \end{aligned}$$

m-function 10.2

The following MATLAB function **lshoot.m** finds the solution of the BVP (10.16) using the linear shooting method. INPUTS are functions *f1(x,u,v)= -p(x)*u-q(x)*v+r(x)* and *f2(x,u,v)= -p(x)*u-q(x)*v*; the end points *a, b*; the boundary conditions α and β; the number of steps *n*. The input functions *f1* and *f2* should be defined as M-files.

```
function lshoot(f1,f2,a,b,alfa,beta,n)
% Solve the 2nd order BVP using the linear shooting method.
% f1= -p(x)*u-q(x)*v+r(x),  f2= -p(x)*u-q(x)*v.
h=(b-a)/n;
y1=alfa;
y2=0;
u=0;
v=alfa;
for i=1:n
% RK of order 4
   x=a+(i-1)*h;
   k1=feval(f1,x,u,v);
   c1=u;
   k2=feval(f1,x+h/2,u+h*k1/2,v+h*c1/2);
   c2=u+h/2*k1;
   k3=feval(f1,x+h/2,u+h*k2/2,v+h*c2/2);
   c3=u+h/2*k2;
   k4=feval(f1,x+h,u+h*k3,v+h*c3);
   c4=u+h*k3;
   u=u+h*(k1+2*k2+2*k3+k4)/6;
   v=v+h*(c1+2*c2+2*c3+c4)/6;
   y1(i+1)=v;
end
u=1;
v=0;
for i=1:n
% RK of order 4
   x=a+(i-1)*h;
   k1=feval(f2,x,u,v);
   c1=u;
   k2=feval(f2,x+h/2,u+h*k1/2,v+h*c1/2);
   c2=u+h/2*k1;
   k3=feval(f2,x+h/2,u+h*k2/2,v+h*c2/2);
   c3=u+h/2*k2;
```

```
    k+4=feval(f2,x+h,u+h*k3,v+h*c3);
    c4=u+h*k3;
    u=u+h*(k1+2*k2+2*k3+k4)/6;
    v=v+h*(c1+2*c2+2*c3+c4)/6;
    y2(i+1)=v;
  end
fprintf('\n')
disp('          Linear shooting method')
fprintf('\n')
disp([' u(b) = ',num2str(y1(n+1)),'  '])
disp([' v(b) = ',num2str(y2(n+1)),'  '])
fprintf('\n')
disp('_____________________________________________________')
disp('          xi          ui          vi          yi      ')
disp('_____________________________________________________')
for i=1:n+1
  x=a+(i-1)*h;
  w=y1(i)+(beta-y1(n+1))/y2(n+1)*y2(i);
  %Write the exact solution if known as s=s(x) otherwise set s='n'.
  s='n';
  if (s=='n')
    fprintf('%6.2f %12.6f %12.6f %12.6f\n',x,y1(i),y2(i),w)
  else
    err=abs(w-s);
    fprintf('%6.2f %12.6f %12.6f %12.6f %12.6f %10.2e\n', x, y1(i),
y2(i), w, s, err)
  end
end
```

EXAMPLE 10.5

Solve the boundary-value problem

$$y'' - \frac{3}{x}y' + \frac{3}{x^2}y = 2x^2e^x, \quad 1 \le x \le 2, \quad y(1) = 0, \quad y(2) = 4e^2$$

using the linear shooting method with $h = 0.1$. Compare the results with the exact solution $y = 2xe^x(x-1)$.

The functions p, q, and r are $p(x) = -3/x$, $q(x) = 3/x^2$, and $r(x) = 2x^2e^x$, respectively. By defining the MATLAB functions

```
function f=f1(x,u,v);
f=3/x*u-3/x.^2*v+2*x.^2*exp(x);
```

```
lshoot('f1','f2',1,2,0,4*exp(2),10)

        Linear shooting method

u(b) = 13.24634
v(b) = 3.000015
```

xi	ui	vi	yi	y(xi)	Error
1.0	0.00000	0.00000	0.00000	0.00000	0
1.1	0.03298	0.11550	0.66091	0.66092	3.75e-006
1.2	0.15838	0.26400	1.59365	1.59366	6.91e-006
1.3	0.42372	0.44850	2.86204	2.86205	9.38e-006
1.4	0.88840	0.67200	4.54181	4.54182	1.11e-005
1.5	1.62568	0.93750	6.72252	6.72253	1.18e-005
1.6	2.72490	1.24801	9.50981	9.50982	1.17e-005
1.7	4.29404	1.60651	13.02798	13.02799	1.04e-005
1.8	6.46274	2.01601	17.42298	17.42298	8.12e-006
1.9	9.38564	2.47951	22.86575	22.86576	4.65e-006
2.0	13.24634	3.00002	29.55622	29.55622	0.00e+000

Table 10.5 The linear shooting method for Example 10.5.

```
function f=f2(x,u,v);
f=3/x*u-3/x.^2*v;
```

and then calling the MATLAB function lshoot, we obtain the results shown in Table 10.5.

EXERCISE SET 10.2

1. Approximate the solution of the following boundary-value problems using the linear shooting method:

   (a) $y'' + xy' + y = x,\ y(0) = 1,\ y(1) = 0,\ h = 0.2.$

   (b) $y'' + (1 - x)y' + xy = x,\ y(0) = 0,\ y(1) = 2,\ h = 0.2.$

   (c) $y'' - \frac{1}{x}y' + \frac{1}{x^2}y = \frac{\ln x}{x^2},\ y(1) = 0,\ y(2) = -2,\ h = 0.2.$

2. Use the nonlinear shooting method to approximate the solution of the following boundary-value problems with $TOL = 10^{-4}$.

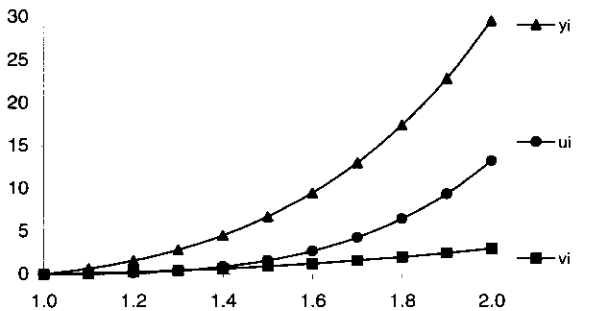


FIGURE 10.2
Numerical solutions ui and vi used to form $y(x) = u(x) + \frac{4e^2 - u(2)}{v(2)} v(x)$ **to solve** $y'' - \frac{3}{x} y' + \frac{3}{x^2} y = 2x^2 e^x$, $n = 10$.

(a) $y'' = xy' + 2y,\ y(0) = -1,\ y(1) = 0,\ h = 0.2.$

(b) $y'' = -\frac{y'^2 + 2y^2}{y},\ y(0) = 1,\ y(\pi/6) = \sqrt{2},\ h = \pi/30.$

(c) $y'' = \frac{1 - y'^2}{y},\ y(0) = 1,\ y(2) = 2,\ h = 0.4.$

3. Use the shooting method to solve the boundary-value problem

$$y'' = y - xy' + 2x + \frac{2}{x}, \quad y(1) = 0, \quad y(2) = 4 \ln 2.$$

Compare with the exact solution $y = 2x \ln x$.

4. **Computer assignment**:
Write a computer program in FORTRAN or C to solve the following boundary-value problem

$$y'' = f(x, y, y'), \quad y(a) = \alpha, \quad y(b) = \beta, \quad b \le x \le b$$

using the nonlinear shooting method.

Test your program to solve the boundary-value problem

$$y'' = x(y')^2, \quad y(0) = \pi/2, \quad y(2) = \pi/4, \quad 0 \le x \le 2.$$

Compare your answer to the actual solution $y = \cot^{-1} \frac{x}{2}$.

APPLIED PROBLEMS FOR CHAPTER 10

1. Under a uniform load, small deflections y of a simple supported beam is given by

$$EI(d^2y/dx^2) = qx(x - L)/2, \quad y(0) = Y(L) = 0$$

where $L = 10$ ft, $EI = 1900$ kip·ft^2, and $q = 0.6$ kip/ft. The beam extends from $(x = 0)$ to $(x = L)$. Find y at every 2 ft by the shooting method.

2. Suppose an 8-lb weight stretches a spring 2 ft beyond its natural length. The weight is then pulled down another $\frac{1}{2}$ ft and released with an initial velocity of 6 ft/sec. The displacement y of the weight at any time t is given by the differential equation

$$\frac{d^2y}{dt^2} + 16y = 0$$

with $y(0) = \frac{1}{2}$ and $y(\pi/2) = \frac{1}{2}$. Approximate the solution of the differential equation in the time interval $[0, \pi/2]$ using the finite difference method described in Section 10.1.

Chapter 11

Eigenvalues and Eigenvectors

The calculation of eigenvalues and eigenvectors is a problem that plays an important part in a large number of applications; both theoretical and practical. They touch most areas in science, engineering, and economics. Some examples are, the solution of the Schrödinger equation in quantum mechanics, the various eigenvalues representing the energy levels of the resulting orbital, the solution of ordinary equations, space dynamics, elasticity, fluid mechanics, and many others.

In this chapter we introduce a new method called the quadratic method (see [11]) for computing the eigenvalues and eigenvectors of a symmetric matrix A. The method is simple and has the advantage of computing the eigenpairs in parallel.

11.1 BASIC THEORY

Le A be a real square, $n \times n$ matrix and let $\mathbf{x}$ be a vector of dimension n. We want to find scalars λ for which there exists a nonzero vector $\mathbf{x}$ such that

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{11.1}$$

When this occurs, we call λ an eigenvalue and $\mathbf{x}$ an eigenvector that corresponds to λ. Together they form an eigenpair $(\lambda\mathbf{x})$ of A. Note that Eqn. (11.1) will have a nontrivial solution only if

$$p(\lambda) = \det(A - \lambda I) = 0 \tag{11.2}$$

The function $p(\lambda)$ is a polynomial of degree n and is known as the characteristic polynomial. The determinant in Eqn. (11.2) can be

written in the form

$$\begin{vmatrix} a_{11}-\lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22}-\lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn}-\lambda \end{vmatrix} = 0. \tag{11.3}$$

It is known that p is an nth degree polynomial with real coefficients and has at most n distinct zeros not necessarily real. Each root λ can be substituted into Eqn. (11.1) to obtain a system of equations that has a nontrivial solution vector $\mathbf{x}$. We now state the following definitions and theorems necessary for the study of eigenvalues. Proofs can be found in any standard texts on linear algebra. See, for example, [5].

DEFINITION 11.1 *The spectral radius $\rho(A)$ of an $n \times n$ matrix A is defined by*

$$\rho(A) = max_{1 \le i \le n} |\lambda_i|$$

where λ_i are the eigenvalues of A.

THEOREM 11.1
The eigenvalues of a symmetric matrix are all real numbers.

THEOREM 11.2
For distinct eigenvalues λ there exists at least one eigenvector $\mathbf{V}$ corresponding to λ.

THEOREM 11.3
If the eigenvalues of an $n \times n$ matrix A are all distinct, then there exists n eigenvectors $\mathbf{V}_j$, for $j = 1, 2, \ldots, n$.

THEOREM 11.4 (Gerschgorin's Circle Theorem)
Let A be an $n \times n$ matrix and let C_i be the disc in the complex plane with center a_{ii} and radius

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

that is, C_i consists of all points z such that

$$|z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

Let D be the union of all the disc C_i for $i = 1, 2, ..., n$; then all the eigenvalues of A lie within D.

As an example consider the matrix

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 3 & 8 & -2 \\ 0 & -2 & -4 \end{bmatrix}$$

All the eigenvalues of this matrix are real since A is symmetric. We have (see Figure 11.1):

C_1 is the disc with center $(2, 0)$ and radius $= 3 + 0 = 3$.
C_2 is the disc with center $(8, 0)$ and radius $= 3 + |-2| = 5$.
C_3 is the disc with center $(-4, 0)$ and radius $= 0 + |-2| = 2$.

The union of these discs is

$$D = [-1, 5] \cup [3, 13] \cup [-6, -2] = [-6, -2] \cup [-1, 13]$$

Thus, the eigenvalues of A must lie within D. In fact, they are

$$\lambda_i = -4.3653, \ 0.8684, \ \text{and } 9.4969.$$

EXAMPLE 11.1
Find the eigenpairs for the matrix

$$\begin{bmatrix} 2 & -3 & 6 \\ 0 & 3 & -4 \\ 0 & 2 & -3 \end{bmatrix}$$

The characteristic equation $\det(A - \lambda I) = 0$ is

$$-\lambda^3 + 2\lambda^2 + \lambda - 2 = 0$$

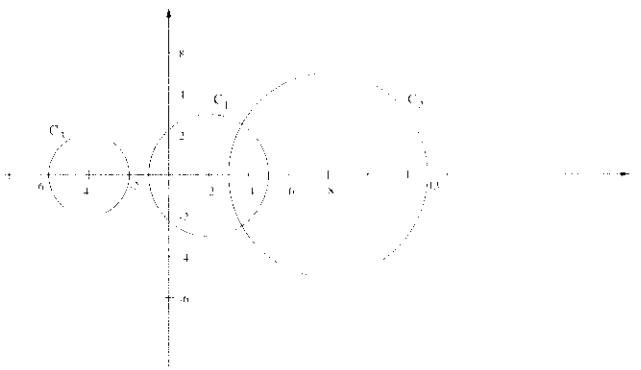


FIGURE 11.1
Gerschgorin's Circle for the matrix A.

The roots of the equation are the three eigenvalues $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = -1$.

To find the eigenvector $\mathbf{x}_1$ corresponding to λ_1, we substitute $\lambda_1 = 1$ to Eqn. (11.1) to get the system of equations

$$\begin{aligned} x_1 - 3x_2 + 6x_3 &= 0 \\ 2x_2 - 4x_3 &= 0 \\ 2x_2 - 4x_3 &= 0 \end{aligned}$$

Since the last two equations are identical, the system is reduced to two equations in three unknowns. Set $x_3 = \alpha$, where α is an arbitrary constant, to get $x_2 = 2\alpha$ and $x_1 = 0$. Hence, by setting $\alpha = 1$, the first eigenpair is

$$\lambda_1 = 1 \quad \text{and } \mathbf{x}_1 = (0, 2, 1)^T.$$

To find $\mathbf{x}_2$, substitute $\lambda_2 = 2$ to Eqn. (11.1) to get the system of equations

$$\begin{aligned} -3x_2 + 6x_3 &= 0 \\ x_2 - 4x_3 &= 0 \\ 2x_2 - 5x_3 &= 0 \end{aligned}$$

The solution of this system is $x_1 = \alpha$, $x_2 = x_3 = 0$. Hence, by setting $\alpha = 1$, the second eigenpair is

$$\lambda_2 = 2 \quad \text{and } \mathbf{x}_2 = (1, 0, 0)^T.$$

Finally, to find $\mathbf{x}_3$ substitute $\lambda_3 = -1$ to Eqn. (11.1) to get the system of equations

$$\begin{aligned} 3x_1 - 3x_2 + 6x_3 &= 0 \\ 4x_2 - 4x_3 &= 0 \\ 2x_2 - 2x_3 &= 0 \end{aligned}$$

The solution of this system is $x_1 = -\alpha$, $x_2 = x_3 = \alpha$. Hence, by setting $\alpha = 1$ the third eigenpair is

$$\lambda_3 = -1 \text{ and } \mathbf{x}_3 = (-1, 1, 1)^T.$$

As one can see from this example, the method is practical when the dimension n is small but for a large n it is difficult to determine the zeros of $p(\lambda)$ and also to find the nonzero solutions of the homogeneous linear system $(A - \lambda I)\mathbf{x} = 0$. There are many techniques for approximating the eigenpairs of a matrix. The most popular ones are the QR and QL algorithms (see [5] for more details). In Section 11.3 we shall introduce a new iterative technique, called the quadratic method, for approximating the eigenpairs of a symmetric matrix but before that we give a brief description of the classical power method.

EXERCISE SET 11.1

1. Find the characteristic polynomials $\rho(\lambda)$ and eigenpairs for the following matrices.

   (a) $A = \begin{bmatrix} 3 & 8 \\ 1 & 1 \end{bmatrix}$ (b) $B = \begin{bmatrix} -2 & 6 \\ 3 & 0 \end{bmatrix}$ (c) $C = \begin{bmatrix} -1 & 3 \\ -2 & 4 \end{bmatrix}$

2. Let $A = \begin{bmatrix} 1 & a+2 \\ a-1 & 4 \end{bmatrix}$

   (a) Find the characteristic polynomial $\rho(\lambda)$ of A.

   (b) Show that the eigenvalues of A are $\lambda_1 = a+3$ and $\lambda_2 = 2-a$.

   (c) Show that the eigenvectors of A are $V_1 = (1, 1)^T$ and $V_2 = (-\frac{a+2}{a-1}, 1)^T$.

3. Let $A = \begin{bmatrix} 3 & 5 & -1 \\ 1 & 4 & -2 \\ 4 & 2 & 1 \end{bmatrix}$

   (a) Use the Gerschgorin Circle Theorem to determine a region containing all the eigenvalues of A.

   (b) Find an interval that contains the spectral radius of A.

4. Let $A = \begin{bmatrix} -2 & 3+m & -2 \\ 1 & 6 & -1 \\ 2+m & 2 & 14 \end{bmatrix}$. If $m > 0$, find all values of m such that the circles in the Gerschgorin Theorem are all disjoint.

5. Let $A = \begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & a+2 \\ b+1 & 1 & 12 \end{bmatrix}$. If $a, b \geq 0$, find the values of a, b, and c such that the center circle in the Gerschgorin Theorem will be tangent to the two others.

6. Find the characteristic polynomial $p(\lambda)$ and eigenpairs for the following matrices:

(a) $A = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 0 & 1 \\ 4 & -4 & 5 \end{bmatrix}$ (b) $B = \begin{bmatrix} 1 & -2 & 2 \\ -2 & 1 & 2 \\ 2 & 2 & 1 \end{bmatrix}$

11.2 THE POWER METHOD

The **power method** is a classical method of use mainly to determine the largest eigenvalue in magnitude, called the **dominant eigenvalue,** and the corresponding eigenvector of the system

$$A\mathbf{x} = \lambda\mathbf{x}.$$

Let A be a real $n \times n$ matrix. Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the distinct eigenvalues of A such that

$$|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n| > 0$$

and $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ be the corresponding eigenvectors. The procedure is applicable if a complete system of n independent eigenvectors exists, even though some of eigenvalues $\lambda_2, ..., \lambda_n$ may not be distinct. Then any eigenvector $\mathbf{x}$ in the space of eigenvectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ can be written as

$$\mathbf{x} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + ... + c_n\mathbf{x}_n \tag{11.4}$$

Premultiplying by A and substituting $A\mathbf{x}_1 = \lambda_1\mathbf{x}_1$, $A\mathbf{x}_2 = \lambda_2\mathbf{x}_2$, etc., Eqn. (11.4) becomes

$$\begin{aligned} A\mathbf{x} &= \lambda_1 c_1\mathbf{x}_1 + \lambda_2 c_2\mathbf{x}_2 + ... + \lambda_n c_n\mathbf{x}_n \\ &= \lambda_1 \left[c_1\mathbf{x}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)\mathbf{x}_2 + ... + c_n\left(\frac{\lambda_n}{\lambda_1}\right)\mathbf{x}_n \right] \end{aligned} \tag{11.5}$$

Premultiplying by A again and simplifying, Eqn. (11.5) becomes

$$A^2\mathbf{x} = \lambda_1^2 \left[c_1\mathbf{x}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^2\mathbf{x}_2 + ... + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^2\mathbf{x}_n \right]$$

$$\cdots$$

$$A^k\mathbf{x} = \lambda_1^k \left[c_1\mathbf{x}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^k\mathbf{x}_2 + ... + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^k\mathbf{x}_n \right] \tag{11.6}$$

$$A^{k+1}\mathbf{x} = \lambda_1^{k+1} \left[c_1\mathbf{x}_1 + c_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1}\mathbf{x}_2 + ... + c_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k+1}\mathbf{x}_n \right] \tag{11.7}$$

As $k \to \infty$, the right-hand sides of (11.6) and (11.7) tend to $\lambda_1^k c_1\mathbf{x}_1$ and $\lambda_1^{k+1} c_1\mathbf{x}_1$, respectively, since $|\lambda_i/\lambda_1| < 1$, $i = 2, ..., n$. The vector $c_1\mathbf{x}_1 + c_2(\lambda_2/\lambda_1)^k\mathbf{x}_2 + ... + c_n(\lambda_n/\lambda_1)^k\mathbf{x}_n$ tends to $c_1\mathbf{x}_1$ which is the eigenvector corresponding to λ_1. The eigenvalue λ_1 is obtained as the ratio of the corresponding components of $A^k\mathbf{x}$ and $A^{k+1}\mathbf{x}$

$$\lambda_1 = \lim_{k\to\infty} \frac{(A^{k+1}\mathbf{x})_r}{(A^k\mathbf{x})_r}, \quad r = 1, 2, ..., n$$

where r denotes the rth component of the vector.

The power method is usually applied as follows. Start with an initial estimate $\mathbf{x}_0$ of the eigenvector $\mathbf{x}$. This estimate is usually chosen to be

$$\mathbf{x}_0 = (1, 1, ..., 1)^T$$

so that the infinity norm $\|\mathbf{x}\|_\infty = \max_{1\le i\le n}\{\,|x_i|\,\}$ of the vector is 1. We then generate the sequence $\{\mathbf{x}_k\}$ recursively, using

$$\begin{aligned} \mathbf{z}_{k-1} &= A\mathbf{x}_{k-1}, \\ \mathbf{x}_k &= \frac{1}{m_k}\mathbf{z}_{k-1} \end{aligned}$$

where m_k is the component of $\mathbf{z}_{k-1}$ with the largest magnitude. If the method converges, the final value of m_k is the required eigenvalue and the final value of $\mathbf{x}_k$ is the corresponding eigenvector. That is,

$$\lim_{k\to\infty} \mathbf{x}_k = \mathbf{x} \quad \text{and} \quad \lim_{k\to\infty} m_k = \lambda.$$

The iteration process is terminated when the infinity norm $\|\mathbf{x}_k - \mathbf{x}_{k-1}\|_\infty$ is less than a specified tolerance.

The method is best illustrated by an example.

EXAMPLE 11.2

Consider the eigenvalue problem

$$\begin{bmatrix} -9 & 14 & 4 \\ -7 & 12 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

where the eigenvalues of A are $\lambda_1 = 5$, $\lambda_2 = 1$, and $\lambda_3 = -2$.

As a first guess, we choose $\mathbf{x}_0 = (1, 1, 1)^T$. Now compute

$$\begin{bmatrix} -9 & 14 & 4 \\ -7 & 12 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 9 \\ 9 \\ 1 \end{bmatrix} = 9 \begin{bmatrix} 1 \\ 1 \\ \frac{1}{9} \end{bmatrix} = m_1 \mathbf{x}_1.$$

We have normalized the vector by dividing through by its largest element. The next iteration yields

$$\begin{bmatrix} -9 & 14 & 4 \\ -7 & 12 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \frac{1}{9} \end{bmatrix} = \frac{49}{9} \begin{bmatrix} 1 \\ 1 \\ \frac{1}{49} \end{bmatrix} = m_2 \mathbf{x}_2.$$

After 10 iterations, the sequence of vectors converges to

$$\mathbf{x} = \left(1, 1, 1.02 \times 10^{-8}\right)^T,$$

and the sequence m_k of constants converges to

$$\lambda = 5.000\,000\,205.$$

A summary of the calculations is given in Table 11.1.

i	$m_i\mathbf{x}_i$
1	$9[1,1,\frac{1}{9}]' = 9\ [1.0, 1.0, 1.11\times 10^{-1}]'$
2	$\frac{49}{9}[1,1,\frac{1}{49}]' = 5.444\,444\ [1.0, 1.0, 2.04\times 10^{-2}]'$
3	$\frac{249}{49}[1,1,\frac{1}{249}]' = 5.081\,633\ [1.0, 1.0, 4.02\times 10^{-3}]'$
4	$\frac{1249}{249}[1,1,\frac{1}{1249}]' = 5.016\,064\ [1.0, 1.0, 8.01\times 10^{-4}]'$
5	$\frac{6249}{1249}[1,1,\frac{1}{6249}]' = 5.003\,2023\ [1.0, 1.0, 1.60\times 10^{-4}]'$
...	...
10	$\frac{97\,656\,249}{19\,531\,249}[1,1,\frac{1}{97656\,249}]' = 5.000\,000\,205\ [1.0, 1.0, 1.02\times 10^{-8}]'$

Table 11.1 Power method for Example 11.2.

The major disadvantage of the power method is that its rate of convergence is slow when the *dominance ratio*

$$r = \left|\frac{\lambda_2}{\lambda_1}\right|$$

of the eigenvalues with the two largest magnitudes is close to one. However, the convergence rate can be accelerated by various strategies. The simplest of these is done by using the power method with **shift**. We know that A and $A - kI$ have the same set of eigenvectors and for each eigenvalue λ_i of A we have, for $A - kI$, the eigenvalue $\lambda_i - k$. That is

$$(A - kI)\mathbf{x} = A\mathbf{x} - k\mathbf{x} = \lambda\mathbf{x} - k\mathbf{x} = (\lambda - k)\mathbf{x}.$$

Therefore, if we subtract k from the diagonal elements of A, then each eigenvalue is reduced by the same factor and the eigenvectors are not changed. The power method can now be used as

$$\mathbf{z}_k = (A - kI)\,\mathbf{x}_k,$$
$$\mathbf{x}_{k+1} = \frac{1}{m_{k+1}}\mathbf{z}_k.$$

For example, if 20 and -18 are the largest eigenvalues in magnitude, the dominance ratio is $r = 0.9$ which is close to 1 and convergence will be relatively slow. However, if we choose $k = \frac{1}{2}(10 + 18) = 14$, then the new eigenvalues are 6 and -32, yielding the dominance ratio of $r = 0.1875$ and a faster rate of convergence. Of course, the choice of k is difficult unless we know, a priori, an estimate of the eigenvalues.

One way of obtaining an estimate of the eigenvalues is by using the Gerschgorin Circle Theorem.

Finally, the eigenvalue with the smallest magnitude can be obtained in a similar way using the **inverse power method**. If λ is an eigenvalue of A and $\mathbf{x}$ is the corresponding eigenvector, then $1/\lambda$ is an eigenvalue of A^{-1} corresponding to the same eigenvector $\mathbf{x}$. Applying the power method to A^{-1} gives an approximation to the eigenvalue of A with the smallest magnitude. The power method can also be used to determine eigenvalues other than the dominant one using *deflation techniques*. For a complete discussion see [15].

EXERCISE SET 11.2

1. Find the dominant eigenpairs of the given matrices

(a) $A = \begin{bmatrix} -17 & 36 & -16 \\ -12 & 25 & -10 \\ 0 & 0 & 3 \end{bmatrix}$ (b) $A = \begin{bmatrix} -4 & 1 & 1 \\ 1 & 5 & -1 \\ 0 & 1 & -3 \end{bmatrix}$

(c) $A = \begin{bmatrix} 2 & 1 & -1 & 3 \\ 1 & 7 & 0 & -1 \\ -1 & 0 & 4 & -2 \\ 3 & -1 & -2 & 1 \end{bmatrix}$ (d) $A = \begin{bmatrix} 1 & 3 & 1 & 0 \\ -1 & 2 & 0 & 3 \\ 4 & -1 & 4 & 1 \\ 2 & 1 & -1 & 8 \end{bmatrix}$

2. Let $(\lambda, \mathbf{x})$ be an eigenpair of A. If $\lambda \neq 0$, show that $(1/\lambda, \mathbf{x})$ is an eigenpair of A^{-1}.

11.3 THE QUADRATIC METHOD

We now describe the quadratic method for computing the eigenpairs of a real symmetric matrix. It is an iterative method which is based on solving quadratic systems.

Let A be an $n \times n$ symmetric matrix. Let $\mathbf{x_i}$ be an eigenvector of A which has 1 in the ith position and λ_i as its corresponding eigenvalue.

Then, the algebraic eigenvalue problem

$$A\mathbf{x_i} = \lambda_i \mathbf{x_i} \tag{11.8}$$

is a nonlinear system of n equations in n unknowns: λ_i, x_i, $i = 1, 2, ..., i-1, i+1, ..., n$.

The ith Eqn. of (11.8) is given by

$$a_{ii} + \sum_{p=1}^{i-1} a_{ip}x_p + \sum_{p=i+1}^{n} a_{ip}x_p = \lambda_i, \tag{11.9}$$

and if $j \neq i$ the jth Eqn. of (11.8) is

$$a_{ji} + \sum_{p=1}^{i-1} a_{jp}\, x_p + \sum_{p=i+1}^{n} a_{jp}\, x_p = \lambda_i\, x_j. \tag{11.10}$$

Using (11.9) and (11.10) we obtain, for $j = 1, ..., i-1, i+1, ..., n$

$$a_{ji} + \sum_{p=1}^{i-1} a_{jp}x_p + \sum_{p=i+1}^{n} a_{jp}x_p = \left[a_{ii} + \sum_{p=1}^{i-1} a_{ip}x_p + \sum_{p=i+1}^{n} a_{ip}x_p\right] x_j. \tag{11.11}$$

For $j = 1, ..., i-1$, (11.11) takes the form

$$\begin{aligned} f_j = a_{ij}x_j^2 &+ \left[a_{ii} - a_{jj} + \sum_{p=1}^{j-1} a_{ip}x_p + \sum_{p=j+1}^{i-1} a_{ip}x_p + \sum_{p=i+1}^{n} a_{ip}x_p\right] x_j \\ &- \left[a_{ji} + \sum_{p=1}^{j-1} a_{jp}x_p + \sum_{p=j+1}^{i-1} a_{jp}x_p + \sum_{p=i+1}^{n} a_{jp}x_p\right] = 0, \end{aligned} \tag{11.12}$$

and for $j = i+1, ..., n$ (11.11) takes the form

$$\begin{aligned} f_j = a_{ij}x_j^2 &+ \left[a_{ii} - a_{jj} + \sum_{p=1}^{i-1} a_{ip}x_p + \sum_{p=i+1}^{j-1} a_{ip}x_p + \sum_{p=j+1}^{n} a_{ip}x_p\right] x_j \\ &- \left[a_{ji} + \sum_{p=1}^{i-1} a_{jp}x_p + \sum_{p=i+1}^{j-1} a_{jp}x_p + \sum_{p=j+1}^{n} a_{jp}x_p\right] = 0. \end{aligned} \tag{11.13}$$

For a fixed i and for $j = 1, 2, ..., i-1, i+1, ..., n$, we define $A_j = a_{ij}$, B_j the coefficient of x_j, and C_j the constant term in Eqns. (11.12) and (11.13).

Hence, finding the eigenvectors of A is reduced to the solution of the quadratic system given by

$$A_j x_j^2 + B_j x_j + C_j = 0, \quad j = 1, 2, \ldots, i-1, i+1, \ldots, n \tag{11.14}$$

We now apply Newton's method to approximate the solution of the system. The $(n-1) \times (n-1)$ Jacobian matrix $J = (q_{jl})$ of the system is defined by

$$(q_{jl}) = \begin{cases} 2A_j x_j + B_j & \text{if } j = l \\ a_{jl} x_j - a_{jl} & \text{if } j \neq l \end{cases}. \tag{11.15}$$

Assuming that the $det(J) \neq 0$, the functional iteration procedure evolves from selecting $\mathbf{x}^{(0)}$ and generating $\mathbf{x}^{(k)}$ for $k \geq 1$, as follows:

- Calculate $\mathbf{F}(\mathbf{x}^{(k-1)})$ and $J(\mathbf{x}^{(k-1)})$
- Solve the $(n-1) \times (n-1)$ linear system $J(\mathbf{x}^{(k-1)})\,\mathbf{z} = -\mathbf{F}(\mathbf{x}^{(k-1)})$
- Generate $\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + \mathbf{z}^{(k-1)}$

where $\mathbf{F} = -(f_1, \ldots, f_{i-1}, f_{i+1}, \ldots, f_n)$.

Once the eigenvectors are obtained, the corresponding eigenvalues are readily obtained from (11.9).

It is well known that Newton's method is generally expected to give a quadratic convergence, provided that a sufficiently accurate starting value is known and $J(\mathbf{x})^{-1}$ exists. To insure an accurate starting value, we introduce a continuation method (see [29]) defined by the family of matrices $H(t)$ such that

$$H(t) = D + tP, \quad 0 \leq t \leq 1 \tag{11.16}$$

where t is a parameter and

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & \ddots & & \vdots \\ \vdots & & & 0 \\ 0 & \cdots & 0 & a_{nn} \end{bmatrix}, \quad tP = \begin{bmatrix} 0 & ta_{12} & \cdots & ta_{1n} \\ ta_{21} & 0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & ta_{n-1,n} \\ ta_{n1} & \cdots & ta_{n,n-1} & 0 \end{bmatrix}$$

Note that the eigenvalues of $H(0) = D$ are given by the diagonal elements of A and that $H(1) = A$. We now consider the problem of finding the eigenpairs of the family of matrices $H(t)$ with $t \in [0, 1]$. Suppose that for each $t \in [0, 1]$ and a given $i = 1, 2, \ldots, n$, $H(t)$ has an

eigenvalue $\lambda_i(t)$ which depends continuously on t. Then $\lambda_i(t)$ describes a curve (solution curve) in R^2 with one end point at the ith diagonal element $\lambda_i(0) = a_{ii}$ of A and the other end point at the ith eigenvalue $\lambda_i(1)$ of A. To obtain $\lambda_i(1)$, we need to define first a partition of the interval $[0, 1]$ by

$$t_k = kh, \; k = 0, 1, .., M, \quad M = 1/h. \tag{11.17}$$

One way of selecting h is by using the Gerschgorin circle theorem. Choose $h = 2^{-s}$ so that all the Gerschgorin circles of the matrix $H(t_1)$ are disjoint (see Figure 11.2), with s given by

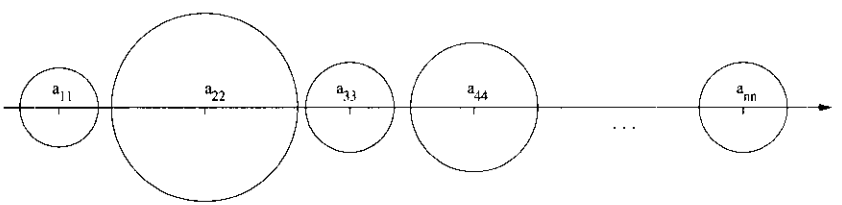


FIGURE 11.2

Gerschgorin circles $\left(a_{ii}, h\sum_{\substack{j=1\\ i\neq j}}^{n} |a_{ij}|\right)$ **of** $H(t_1)$.

$$INT\left(\frac{\ln(r/d)}{\ln 2} + 2\right), \quad r = \max_i \left(\sum_{\substack{j=1\\ i\neq j}}^{n} |a_{ij}|\right), \text{ and } d = \min_{\substack{1\leq i,j\leq n\\ i\neq j}} |a_{ii} - a_{jj}| \tag{11.18}$$

provided that $a_{ii} \neq a_{jj}$. Here $INT(x)$ represents the value of x truncated to an integer. Thus, given h, the method consists of finding the sequence of eigenpairs of $H(t_0), H(t_1), ..., H(t_M)$ obtained by solving, for each t_k, the nonlinear system (11.12) and (11.13) using Newton's method.

The steps for finding the starting value for the application of Newton's method are as follows:

- At $t_0 = 0$, the eigenvalues of $H(0)$ are known and given by the diagonal elements of A and the corresponding eigenvectors are

$$\mathbf{x}^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, ..., x_n^{(0)}\right) = \left(\begin{bmatrix}1\\0\\\vdots\\0\end{bmatrix}, \begin{bmatrix}0\\1\\\vdots\\0\end{bmatrix}, ..., \begin{bmatrix}0\\0\\\vdots\\1\end{bmatrix}\right).$$

Here, $x_i^{(k)}$ denotes the ith eigenvector of the matrix $H(t_k)$.

- At $t = t_1$, Newton's method is applied to the matrix $H(t_1)$ using $\mathbf{x}^{(0)}$ as a starting value to get $\mathbf{x}^{(1)}$.
- At $t = t_2$, the two-point interpolation formula

$$\mathbf{y}^{(2)} = 2\mathbf{x}^{(1)} - \mathbf{x}^{(0)} \tag{11.19}$$

  is used to find the starting value $\mathbf{y}^{(2)}$. Here $\mathbf{x}^{(0)}$ and $\mathbf{x}^{(1)}$ are the eigenvectors of $H(t_0)$ and $H(t_1)$, respectively.
- For $t_k > t_2$, the previous three eigenvectors, i.e., $\mathbf{x}^{(k-1)}$, $\mathbf{x}^{(k-2)}$, $\mathbf{x}^{(k-3)}$ of $H(t_{k-1})$, $H(t_{k-2})$, $H(t_{k-3})$, respectively, are used together with the following three-point interpolation formula

$$\mathbf{y}^{(k)} = 3\mathbf{x}^{(k-1)} - 3\mathbf{x}^{(k-2)} + \mathbf{x}^{(k-3)}, \quad k \geq 3 \tag{11.20}$$

  to find the starting value $\mathbf{y}^{(k)}$ for Newton's method that is needed to compute the eigenpairs of $H(t_k)$.

Hence, the method consists of finding the eigenpairs of the sequence of matrices $H(t_0), H(t_1), ..., H(t_M)$ by solving at each step quadratic nonlinear systems using formula (11.20) to compute the starting values.

EXAMPLE 11.3

Use the quadratic method to find the eigenpairs for the matrix

$$\begin{bmatrix} 3 & 2 & -2 \\ 2 & -4 & 2 \\ -2 & 2 & 6 \end{bmatrix}$$

For this matrix we have $h = 0.25$ obtained using formula 11.18. We begin by considering the following sequence of matrices

$$\tilde{A}_k = D + t_k P = \begin{bmatrix} 3 & 2t_k & -2t_k \\ 2t_k & -4 & 2t_k \\ -2t_k & 2t_k & 6 \end{bmatrix}$$

where $t_k = kh = 0.25k, \quad k = 0, 1, ..., 4.$

The algorithm of the quadratic method for finding the first eigenpair $(\lambda_1, \mathbf{x}_1)$ where $\mathbf{x}_1 = [1, x_2, x_3]^T$ proceeds as follows:

From Eqn. (11.1) we have

$$\begin{bmatrix} 3 & 2t_k & -2t_k \\ 2t_k & -4 & 2t_k \\ -2t_k & 2t_k & 6 \end{bmatrix} \begin{bmatrix} 1 \\ x_2 \\ x_3 \end{bmatrix} = \lambda_1 \begin{bmatrix} 1 \\ x_2 \\ x_3 \end{bmatrix}$$

which can be written as

$$\begin{aligned} 3 + 2x_2t_k - 2x_3t_k &= \lambda_1 \\ 2t_k - 4x_2 + 2x_3t_k &= \lambda_1 x_2 \\ -2t_k + 2x_2t_k + 6\,x_3 &= \lambda_1 x_3 \end{aligned} \tag{11.21}$$

Substitute λ_1 into the last two equations to get the following nonlinear system in two unknowns x_2 and x_3

$$\begin{aligned} -2x_2^2t_k + (2x_3t_k - 7)x_2 + 2x_3t_k + 2t_k &= 0 \\ 2x_3^2t_k - (2x_2t_k - 3)x_3 + 2x_2t_k - 2t_k &= 0 \end{aligned} \tag{11.22}$$

We now solve system (11.22) for different values of t_k, using Newton's method with a tolerance of 10^{-6}.

Step 1: To find the eigenpair of $\tilde{A}_0$, set $t_0 = 0$ in (11.22) to get the following eigenvector

$$\mathbf{x}_1^{(0)} = [1, 0, 0]^T$$

and then its corresponding eigenvalue

$$\lambda_1^{(0)} = a_{11} = 3.$$

Step 2: To find the eigenpair of $\tilde{A}_1$, set $t_1 = 0.25$ in (11.22) to get the following system

$$\begin{aligned} -0.5x_2^2 + (0.5x_3 - 7)x_2 + 0.5x_3 + 0.5 &= 0 \\ 0.5x_3^2 - (0.5x_2 - 3)x_3 + 0.5x_2 - 0.5 &= 0. \end{aligned}$$

By applying Newton's method with

$$\mathbf{y}_1^{(1)} = \mathbf{x}_1^{(0)} = [1, 0, 0]^T$$

as the starting value, we obtain after 3 iterations the following eigenvector:

$$\mathbf{x}_1^{(1)} = [1, 0.0826308129, 0.15116808965]^T.$$

The corresponding eigenvalue is obtained by substituting $\mathbf{x}_1^{(1)}$ into Eqn. (11.21), that is:

$$\lambda_1^{(1)} = 3 + 0.5(0.0826308129) - 0.5\,(0.15116808965) = 2.96573136162.$$

Step 3: To find the eigenpair of $\tilde{A}_2$, set $t_2 = 0.5$ in (11.22) to get the following system

$$\begin{aligned} -x_2^2 + (x_3 - 7)x_2 + x_3 + 1 &= 0 \\ x_3^2 - (x_2 - 3)x_3 + x_2 - 1 &= 0. \end{aligned}$$

By applying Newton's method with

$$\mathbf{y}_1^{(2)} = 2\mathbf{x}_1^{(1)} - \mathbf{x}_1^{(0)} = [1, -.834\,738\,374\,2, -.697\,663\,820\,6]^T$$

as the starting value, we obtain, after 3 iterations, the following eigenvector

$$\mathbf{x}_1^{(2)} = [1,\ 0.1828779200, 0.2651074706]^T.$$

The corresponding eigenvalue is obtained by substituting $\mathbf{x}_1^{(2)}$ into Eqn. (11.21), that is:

$$\lambda_1^{(2)} = 3 + (0.1828779200) - (0.2651074706) = 2.9177704494.$$

Step 4: Finally, to find the eigenpair of $\tilde{A}_3$, set $t_3 = 0.75$ in (11.22) to get the following system

$$\begin{aligned} -1.5x_2^2 + (1.5x_3 - 7)x_2 + 1.5x_3 + 1.5 &= 0 \\ 1.5x_3^2 - (1.5x_2 - 3)x_3 + 1.5x_2 - 1.5 &= 0. \end{aligned}$$

By applying Newton's method with

$$\mathbf{y}_1^{(3)} = 3\mathbf{x}_1^{(2)} - 3\mathbf{x}_1^{(1)} + \mathbf{x}_1^{(0)} = [1, 1.300\,741\,321, 1.341\,818\,143]^T$$

as the starting value, we obtain, after 3 iterations, the following eigenvector

$$\mathbf{x}_1^{(3)} = [1, 0.2915541714, 0.3450029259]^T.$$

The corresponding eigenvalue is obtained by substituting $\mathbf{x}_1^{(3)}$ into Eqn. (11.21), that is:

$$\lambda_1^{(3)} = 3 + 1.5(0.2915541714) - 1.5\,(0.3450029259) = 2.919\,826\,868.$$

Step 5: To find the eigenpair of $\tilde{A}_4 = A$, set $t_4 = 1$ in (11.22) to get the following system

$$\begin{aligned} -2x_2^2 + (2x_3 - 7)x_2 + 2x_3 + 2 &= 0 \\ 2x_3^2 - (2x_2 - 3)x_3 + 2x_2 - 2 &= 0. \end{aligned}$$

By applying Newton's method with

$$\mathbf{y}_1^{(4)} = 3\mathbf{x}_1^{(3)} - 3\mathbf{x}_1^{(2)} + \mathbf{x}_1^{(1)} = [1, 0.408\,659\,567\,1, 0.390\,854\,455\,9]^T$$

as the starting value, we obtain, after 3 iterations, the following eigenvector

$$\mathbf{x}_1^{(4)} = \mathbf{x}_1 = [1, 0.4, 0.4]^T.$$

The corresponding eigenvalue is obtained by substituting $\mathbf{x}_1^{(4)}$ into Eqn. (11.21), that is

$$\lambda_1^{(4)} = \lambda_1 = 3 + 2(0.4) - 2\,(0.4) = 3.0.$$

The two other eigenpairs of A can be obtained in a similar manner. Figure (11.3) shows the solution curves of A.

It is important to note the unique feature of this method which is that all the eigenpairs can be found independently of each other, hence, the suitability of this procedure for parallel processing. The main disadvantage of the method is that it requires the solution of an $(n-1) \times (n-1)$ quadratic system at each step.

EXAMPLE 11.4

Use the quadratic method to find the eigenpairs for the matrix

$$A = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 4 & 2 \\ 0 & 2 & 3 \end{bmatrix}$$

Note that in this example $a_{11} = a_{33}$, which means that the formula in (11.18) cannot be applied. The following modifications must be made to account for this case:

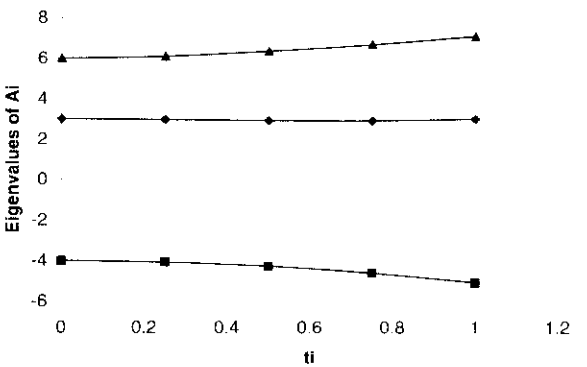


FIGURE 11.3
Solution curves obtained from the eigenvalues of $\mathbf{A}_i$ at $t_i, i = 0, ..., 4$. for Example 11.3.

We apply a "relative perturbation", say $\varepsilon = 10^{-3}$, to the identical diagonal elements of matrix A to get the following new matrix:

$$B = \begin{bmatrix} 3 & 1 & 0 \\ 1 & 4 & 2 \\ 0 & 2 & 3(1+\varepsilon) \end{bmatrix}$$

whose diagonal elements are distinct. We now apply the quadratic method to B as in Example 11.3 and use the resulting eigenvectors of B as the starting values for Newton's method with $h = 1$ to find the eigenpairs of A. The result is:

By applying the quadratic method we obtain the following eigenvectors of B:

$$\mathbf{x}_B = \left[\begin{pmatrix} 1.0 \\ -1.7898267750 \\ 1.9966533298 \end{pmatrix}, \begin{pmatrix} 0.3581371370 \\ 1.0 \\ 0.7170446750 \end{pmatrix}, \begin{pmatrix} -1.9988008642 \\ -.0011998560 \\ 1.0 \end{pmatrix} \right]$$

Using $\mathbf{x}_B$ as the starting values for Newton's method, we apply the quadratic method with $h = 1$ to get the following eigenpairs of A:

$\lambda_1 = 1.2087121525$ and $\mathbf{x}_1 = [1.0, -1.7912878475, 2.0]^T$,

$\lambda_2 = 5.7912878475$ and $\mathbf{x}_2 = [0.3582575695, 1.0, 0.7165151390]^T$,

$\lambda_3 = 3.0000000000$ and $\mathbf{x}_3 = [-2.0, 0.0, 1.0]^T$.

The MATLAB function for finding the eigenvalues and eigenvectors of a given square matrix A is

```
>> [V,D]=eig(A)
```

When called, the function returns a diagonal matrix D of eigenvalues and a full matrix V whose columns are the corresponding eigenvectors. For example, if we call this function to find the eigenpairs of the matrix given in Example 11.4, we get the output shown in Table 11.2. These results are in complete agreement with the ones obtained by using the quadratic method.

```
» A=[3 1 0;1 4 2;0 2 3];
» [V,D]=eig(A)

V =

   0.8944   0.3490   0.2796
   0.0000  -0.6252   0.7805
  -0.4472   0.6981   0.5592


D =

   3.0000        0        0
        0   1.2087        0
        0        0   5.7913
```

Table 11.2 Eigenvalues and eigenvectors of A for Example 11.4.

EXERCISE SET 11.3

1. Use the quadratic method to find the eigenpairs of the following matrices.

(a) $A = \begin{bmatrix} 3 & -4 & 6 \\ -4 & 9 & 2 \\ 6 & 2 & 4 \end{bmatrix}$ (b) $A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 0 & 3 \\ 2 & 3 & 0 \end{bmatrix}$

(c) $A = \begin{bmatrix} -7 & 0 & 6 \\ 0 & 5 & 0 \\ 6 & 0 & 2 \end{bmatrix}$ (d) $A = \begin{bmatrix} -1 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & -1 \end{bmatrix}$

(e) $A = \begin{bmatrix} -5 & 1 & 1 & 0 \\ 1 & -8 & 2 & 0 \\ 1 & 2 & -6 & 1 \\ 0 & 0 & 1 & -3 \end{bmatrix}$

2. Find the largest eigenvalue of the matrix $A = \begin{bmatrix} -1 & 0 & 3 \\ 0 & 3 & 0 \\ 3 & 0 & -1 \end{bmatrix}$

3. Consider the system of differential equations

$$X'(t) = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 0.1 & 3 \\ 2 & 3 & 0.2 \end{bmatrix} X(t)$$

The general solution to this problem is

$$X(t) = Ae^{\lambda_1 t}X_1 + Be^{\lambda_2 t}X_2 + Ce^{\lambda_3 t}X_3$$

where (λ_1, X_1), (λ_2, X_2), and (λ_3, X_3) are the eigenpairs of the coefficient matrix of the system.

Find the general solution of the system using the quadratic method.

4. Use the quadratic method to find the eigenpairs of the following tridiagonal matrix:

$$A = \begin{bmatrix} 5 & -2 & 0 & 0 & 0 \\ -2 & 5 & -2 & 0 & 0 \\ 0 & -2 & 5 & -2 & 0 \\ 0 & 0 & -2 & 5 & -2 \\ 0 & 0 & 0 & -2 & 5 \end{bmatrix}$$

5. The Hilbert matrix A of order n is defined by

$$a_{ij} = \frac{1}{i+j-1}, \quad i, j = 1, 2, \ldots, n$$

Use the quadratic method to find the eigenvalues of A for $n = 4, 8$.

11.4 EIGENVALUES FOR BOUNDARY-VALUE PROBLEMS

Eigenvalue problems arise in many branches of applied mathematics, and in particular, in the solutions of differential equations such as problems in the fields of elasticity and vibration. Consider, for example, the homogeneous second-order differential equation

$$\frac{d^2y}{dx^2} + k^2\, y = 0, \quad 0 \le x \le L \tag{11.23}$$

subject to the boundary conditions

$$y(0) = 0 \quad \text{and} \quad y(L) = 0 \tag{11.24}$$

where k^2 is a parameter. Here, $y(x)$ denotes the deflection of a thin elastic column of length L when a constant vertical compressive force P is applied to its top, as shown in Figure 11.4. The parameter k is given by $k = P/EI$, where E is Young's modulus of elasticity and I is the moment of inertia of a cross-section about a vertical line through its centroid.

Eqn. (11.23), along with the boundary conditions (11.24), falls into a special class of boundary-value problems known as eigenvalue problems. The values of k^2 for which the boundary-value problem has nontrivial solutions are called the eigenvalues. The general solution of (11.23) is

$$y(x) = c_1 \cos(kx) + c_2 \sin(kx). \tag{11.25}$$

If we apply the boundary conditions (11.24) to the general solution we get:

The condition $y(0) = 0$ yields $c_1 = 0$, but $y(L) = 0$ implies

$$c_2 \sin(kx) = 0.$$

If $c_2 = 0$, then necessarily $y = 0$ which is called the trivial solution. However, if $c_2 \neq 0$, then $\sin(kx) = 0$. The last condition is true only for the certain values of k:

$$k_n = \pm \frac{n\pi}{L}, \quad n = 1, 2, 3, \ldots.$$

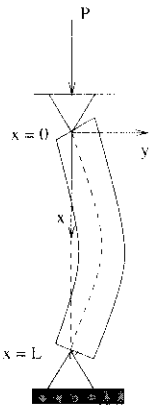


FIGURE 11.4
Bucking of a thin vertical column.

These are the eigenvalues for the equation, and the corresponding eigenfunctions are

$$y_n = c_2 \sin(\frac{n\pi}{L}x), \quad n = 1, 2, 3, \ldots.$$

These eigenvalues are important for the vibration problem because they give the natural frequency of the system. That is, if the external load P is applied very near to these frequencies, resonance causes an amplification of the motion.

We now describe a numerical solution to the eigenvalue problem (11.23) and (11.24).

Let us divide the interval $[0, L]$ into n equal subintervals defined by

$$x_i = hi, \quad h = L/n, \quad i = 0, 1, \ldots, n.$$

By replacing the second derivative in (11.23) with a central difference approximation, we obtain the difference equation

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2} + k^2 y_i = 0$$

or

$$-y_{i-1} + 2y_i - y_{i+1} = h^2 k^2 y_i, \quad i = 1, 2, \ldots, n-1. \tag{11.26}$$

In matrix form Eqn. (11.26) can be written as

$$A\mathbf{y} = \lambda \mathbf{y}$$

where

$$A = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y \\ y_{n-1} \end{bmatrix} \quad \text{and} \quad \lambda = h^2k^2. \tag{11.27}$$

The solutions of the system of equations (11.26) are given by the eigenpairs of A which can be obtained by using the quadratic method described in the previous section. For example, let $L = 1$ and $n = 6$ or $h = 1/6$. The eigenvalues of the 5×5 matrix A are

$$\lambda_1 = 0.2679, \ \lambda_2 = 1.0, \ \lambda_3 = 2.0, \ \lambda_4 = 3.0, \ \text{and } \lambda_5 = 3.7321$$

which give these values for k

$$k_1 = \pm 3.1058, \ k_2 = \pm 6.0, \ k_3 = \pm 8.4853,$$
$$k_4 = \pm 10.3923, \ k_5 = \pm 11.5911.$$

The analytical values for k are given by

$$k_1 = \pm 3.1415, \ k_2 = \pm 6.2832, \ k_3 = \pm 9.4248,$$
$$k_4 = \pm 12.5664, \ \text{and } k_5 = \pm 15.7080.$$

We did not get a good approximation of the values of k due to the small number n of the subintervals. To get a better approximation, one needs to choose a larger number of subintervals.

EXERCISE SET 11.4

1. Estimate the eigenvalues of the boundary-value problem

$$y'' - 2y' + 4k^2y = 0, \quad y(0) = 0, \quad y(1) = 0.$$

using $h = 0.25$.

11.5 BIFURCATIONS IN DIFFERENTIAL EQUATIONS

Systems of ordinary differential equations, modeling physical or biological problems, typically contain parameters, which in a given situation must be assigned definite numerical values. The behavior of the solutions may, however, be radically affected by the values of the parameters chosen. Consider for example the logistic or Verhulst population model

$$\frac{dN}{dt} = rN - aN^2$$

which is of interest only for $N \geq 0$. We assume $a > 0$ has been chosen, and consider the dependence of the solution on the parameter r. The fixed points or equilibrium solutions are given by

$$rN - aN^2 = 0.$$

If $r \leq 0$, the only fixed point is $N = 0$, and it is stable. In fact, for every initial condition

$$N(0) = N_0 > 0,$$

the solution satisfies

$$\lim_{t \to \infty} N(t) = 0.$$

If $r > 0$, there are two fixed points, $N = 0$ and $N = r/a$. The fixed point $N = 0$ becomes unstable and the fixed point $N = r/a$ is stable. The fixed point is said to bifurcate at $r = 0$. If we think of the fixed point, N, as a function of r so that $N = N(r)$, then $(0, 0)$ is called a **bifurcation point**. We illustrate the behavior described with the diagram shown in Figure 11.5.

The philosophy for the general case is quite similar. For the sake of simplicity in presentation, we shall restrict ourselves to the case of two differential equations in two unknown functions, $x(t)$ and $y(t)$, and consider the autonomous system

$$\frac{dx}{dt} = f(x, y, r) \quad \text{and} \quad \frac{dy}{dt} = g(x, y, r), \tag{11.28}$$

where r is a parameter. However, the discussion works for any number of differential equations. Moreover, the second-order differential equation

$$\frac{d^2u}{dt^2} = F\left(u, \frac{du}{dt}, r\right)$$

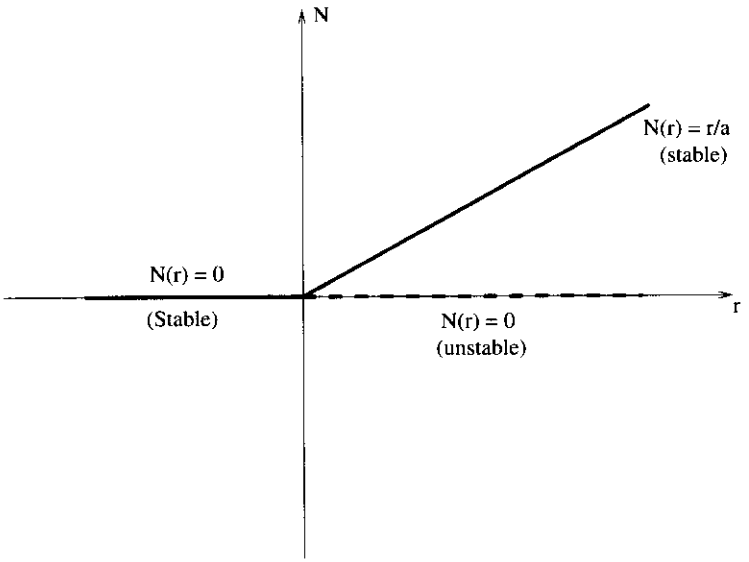


FIGURE 11.5
Bifurcation point.

can be reduced to a form equivalent to (11.28) by setting $x = u$ and $y = du/dt$ to obtain

$$\frac{dx}{dt} = y \quad \text{and} \quad \frac{dy}{dt} = F(x, y, r).$$

The fixed points of (11.28) are obtained by solving the nonlinear system

$$f(x, y, r) = 0 \quad \text{and} \quad g(x, y, r) = 0 \tag{11.29}$$

to obtain the solutions $(x^*(r), y^*(r))$ which depend on r. Intuitively, for a fixed r, (x^*, y^*) is stable if a solution which starts sufficiently close to (x^*, y^*) remains close to (x^*, y^*) and it is unstable if it is forced to leave the given neighborhood no matter how close one starts to (x^*, y^*). Now form the matrix

$$J(x^*, y^*, r) = \begin{bmatrix} f_x(x^*, y^*, r) & f_y(x^*, y^*, r) \\ g_x(x^*, y^*, r) & g_y(x^*, y^*, r) \end{bmatrix}. \tag{11.30}$$

Let us note that for a given r, there may be several distinct values of x and y which satisfy (11.29). Let us suppose that $\overline{r}$, $f(\overline{x}, \overline{y}, \overline{r}) = 0$ and $g(\overline{x}, \overline{y}, \overline{r}) = 0$. If

$$\det[J(\overline{x}, \overline{y}, \overline{r})] \neq 0,$$

then in a neighborhood of $\overline{r}$, there are functions $x^*(r)$ and $y^*(r)$ such that

$$x^*(\overline{r}) = \overline{x} \quad \text{and} \quad y^*(\overline{r}) = \overline{y}$$

by the implicit function theorem. Now we are seeking values of r where branching can occur, and at this branching there is often a change in the stability of the fixed point. A necessary condition for branching to occur is, therefore,

$$\det\left[J\left(x^*, y^*, r\right)\right] = 0. \tag{11.31}$$

EXAMPLE 11.5

Consider the case of a frictionless bead of mass m moving on a hoop of radius r which, is itself, twirled at a rate of constant circular frequency $\omega \geq 0$. See Figure 11.6.

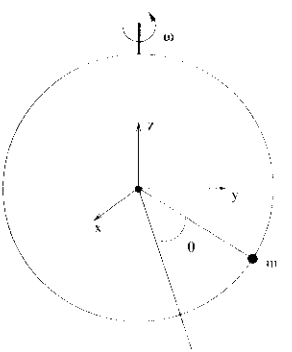


FIGURE 11.6
Frictionless bead.

The position of m at any time, t, is

$$\begin{aligned} x &= R\cos\omega t\sin\theta \\ y &= R\sin\omega t\sin\theta \\ z &= -R\cos\theta \end{aligned}$$

Its kinetic energy, K, is

$$K = \frac{mR^2}{2}\left(\omega^2\sin^2\theta + \theta'^2\right)$$

and its potential energy, V, is

$$V = mgR\,(1 - \cos\theta)\,.$$

The equation of motion is

$$mR^2\frac{d^2\theta}{dt^2} = mR^2\omega^2\sin\theta\cos\theta - mgR\sin\theta.$$

Divide the equation by mR^2 and let $u = \theta$ and $v = d\theta/dt$ to obtain the system

$$\frac{du}{dt} = v \quad \text{and} \quad \frac{dv}{dt} = \omega^2\sin u\cos u - \frac{g}{R}\sin u. \tag{11.32}$$

The equilibrium points in the uv–phase plane satisfy $v = 0$ and

$$\omega^2\sin u\cos u - \frac{g}{R}\sin u = \omega^2\sin u\left(\cos u - \frac{g}{\omega^2 R}\right) = 0.$$

Now $\sin u = 0$ implies $u = 0$ or $u = \pi$. One can, of course, add multiples of 2π, but that is irrelevant. The point $(0,0)$ corresponds to the point on the very bottom of the hoop and the point $u = \pi$ corresponds to the case where the bead sits at the very top of the hoop. Whether or not there are other fixed points depends upon the existence of solutions to $\cos u - g/\omega^2 R = 0$ and this in turn depends upon how fast the hoop spins. If the hoop spins slowly, then ω is small and $g/\omega^2 R > 1$ so there are no more fixed points. While if ω is large and $g/\omega^2 R < 1$ then an additional fixed point corresponds to $\theta = \arccos\,(g/\omega^2 R)$. The Jacobian matrix of the system (11.32) is

$$A\,(u,v,\omega) = \begin{bmatrix} 0 & 1 \\ \omega^2\left\{(\cos^2 u - \sin^2 u) - \frac{g}{\omega^2 R}\cos u\right\} & 0 \end{bmatrix}$$

$$A\,(0,0,\omega) = \begin{bmatrix} 0 & 1 \\ \omega^2 - g/R & 0 \end{bmatrix}$$

$$A\,(\pi,0,\omega) = \begin{bmatrix} 0 & 1 \\ \omega^2 + g/R & 0 \end{bmatrix}$$

$$A\left(\cos^{-1}\frac{g}{\omega^2 R}, 0, \omega\right) = \begin{bmatrix} 0 & 1 \\ (g^2/\omega^2 R^2) - \omega^2 & 0 \end{bmatrix}$$

$\det[A\,(\pi,0,\omega)] = -\,(\omega^2 + g/R)$ is never zero, so this fixed point will have no bifurcations from it. when $\omega = \sqrt{g/R}$ then $\det\,[A\,(0,0,\omega)] =$

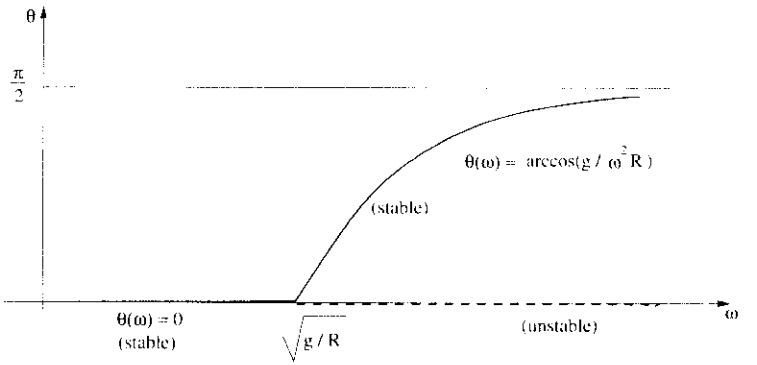


FIGURE 11.7
Bifurcation point.

$-(\omega^2 - g/R) = 0$ and the bottom can bifurcate. The bifurcation diagram for the fixed point θ as a function of ω, is shown in Figure 11.7.

The investigation of stability will take us too far a field in this short introduction; and we shall limit ourselves to determining stability experimentally by taking many values of ω (see exercises below).

EXERCISE SET 11.5

1. Find the fixed points and draw bifurcation diagrams for the following one-dimensional problems. Stability is not an issue here.

   (a) $x^3 + 2x^2 + rx = 0$

   (b) $rx - x^3 = 0$

   (c) $r^2x - x^3 = 0$

   (d) $r^2x + x^3 = 0$

2. Consider the problem of the bead on the hoop of Example 11.5. Let $g = 1$ and $r = 1$. In this problem we will take $\theta'(0) = 0$.

   (a) Let $\omega = 0.5$. Make phase plane plots corresponding to the initial values $\theta(0) = 0.1$ and $\theta(0) = 3.14$. On the basis of these plots, what can you say about the stability of the point $(0, 0)$ and $(\pi, 0)$?

(b) Let $\omega = 1.5$. Make phase plane plots corresponding to the initial values $\theta(0) = 0.1$ and $\theta(0) = 3.14$. Compare these plots with those of part (a).

(c) Let $\omega = 10$. Make phase plane plots corresponding to the initial values $\theta(0) = 0.1$ and $\theta(0) = 3.14$. Compare these plots with those of part (b).

3. Consider, again, the problem of the bead on the hoop of Example 11.5. Assume that friction forces act and that the equation of motion is

$$mR^2\frac{d^2\theta}{dt^2} = -mR^2c\frac{d\theta}{dt} + mR^2\omega^2\sin\theta\cos\theta - mgR\sin\theta.$$

Find the fixed points and determine if they bifurcate with ω. Let $c = 2$ and answer parts (b) and (c) of Problem 2 for this case.

4. Consider the system of differential equations

$$\frac{dx}{dt} = rx + y$$
$$\frac{dy}{dt} = -\left(x + ry + rx^3\right)$$

with $r > 0$. Show that the origin is the only fixed point where $r < 1$. Find the fixed points for $r > 1$. Show that $r = 1$ is a critical point, that is a possible bifurcation point. Choose a number of initial values $(x(0), y(0))$ for different values of r and determine the stability of the origin and the other fixed points experimentally. What happens when $r = 0$?

APPLIED PROBLEMS FOR CHAPTER 11

1. Consider the Lorentz system of differential equations

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = -\beta + xy$$

where σ, β, r are positive constants. In this problem, σ and β will be regarded as fixed but we will assign different values to r, although, in any given example, r will be fixed. In the numerical examples, we will take $\sigma = 10$ and $\beta = 8/3$, which are traditional values.

(a) Find the fixed points for the system.

Note: $(0, 0, 0)$ will always be a fixed point. You will find that it is the only fixed point when $r < 1$ but there will be others when $r > 1$.

(b) Show experimentally that the origin is stable when $r < 0$ and unstable when $r > 1$.

(c) Show that $r = 1$ is a bifurcation point.

(d) Make three-dimensional plots in the phase plane of the solutions to the Lorentz system for $r = 0.5, 2, 10, 28, 56, 100$. Choose initial conditions close to the origin.

Chapter 12

Partial Differential Equations

This chapter presents a brief introduction to some techniques available for approximating the solution to partial differential equations (PDEs) of the form

$$A\frac{\partial^2}{\partial x^2}u(x,y)+B\frac{\partial^2}{\partial x\,\partial y}u(x,y)+C\frac{\partial^2}{\partial y^2}u(x,y)=f\left(x,y,u,\frac{\partial u}{\partial x},\frac{\partial u}{\partial y}\right). \tag{12.1}$$

Eqn. (12.1) are classified into three categories depending on the values of the coefficients A, B, and C:

If	$B^2-4AC>0,$	the equation is **hyperbolic**.
If	$B^2-4AC=0,$	the equation is **parabolic**.
If	$B^2-4AC<0,$	the equation is **elliptic**.

In many applications, the y variable refers to the time and when that is the case, we write t instead of y.

In the next sections, we will discuss solution methods for the parabolic partial differential equation given by

$$\alpha\frac{\partial^2 u}{\partial x^2}=\frac{\partial u}{\partial t},\quad \alpha>0, \tag{12.2}$$

and known as the heat or diffusion equation.

Another equation for which we discuss solution methods is the hyperbolic partial differential equation given by

$$\alpha^2\frac{\partial^2 u}{\partial x^2}=\frac{\partial^2 u}{\partial t^2}, \tag{12.3}$$

and known as the wave equation.

Finally, we will discuss solution methods for the elliptic partial differential equation given by

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0 \tag{12.4}$$

and known as the Laplace's equation.

12.1 PARABOLIC EQUATIONS

We begin our discussion with the one-dimensional PDEs of parabolic type. Consider Eqn. (12.2)

$$\alpha \frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \quad 0 < x < L, \quad 0 < t \leq T \tag{12.5}$$

subject to the boundary conditions

$$u(0,t) = 0, \quad u(L,t) = 0, \quad 0 \leq t \leq T. \tag{12.6}$$

and the initial conditions

$$u(x,0) = f(x), \quad 0 \leq x \leq L. \tag{12.7}$$

Here, $u(x,t)$ denotes the temperature at any time t along a thin, long rod of length L in which heat is flowing as illustrated in Figure 12.1. We assume that the rod is of homogeneous material and has cross-sectional area A that is constant throughout the length of the rod. The rod is laterally insulated along its entire length. The constant α in Eqn. (12.5) is determined by the thermal properties of the material and is a measure of its ability to conduct heat.

12.1.1 Explicit methods

One important approach to approximate the solution to Eqns. (12.5)-(12.7) is the finite difference method. In order to approximate the solution for our problem, a network of grid points is first established throughout the rectangular region

$$R = \{(x,t) \,|\, 0 \leq x \leq L, 0 \leq t \leq T\}$$

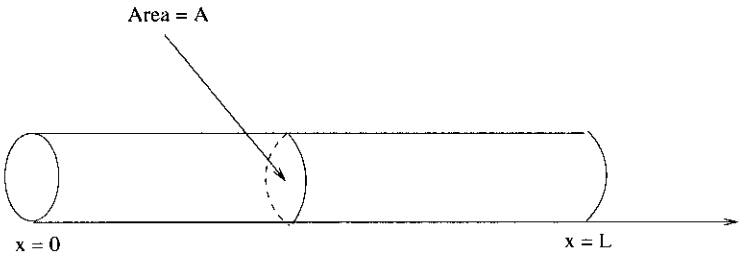


FIGURE 12.1
One-dimensional rod of length L.

as shown in Figure 12.2. We partition R by dividing the interval $[0, L]$ into n equal subintervals, each of length $h = L/n$ and the interval $[0, T]$ into m equal subintervals, each of length $k = T/m$. The corresponding points of the intervals $[0, L]$ and $[0, T]$ are denoted by x_i, for $i = 0, ..., n$ and t_j, for $j = 0, ..., m$, respectively. The points (x_i, t_j) are called the **mesh** or **grid** points and are defined by

$$x_i = hi, \quad t_j = kj \quad \text{for } i = 0, ..., n, \text{ and } j = 0, ..., m.$$

The approximate solution of $u(x, t)$ at the mesh point (x_i, t_j) is denoted by u_i^j and the true solution is denoted by $u(x_i, t_j)$.

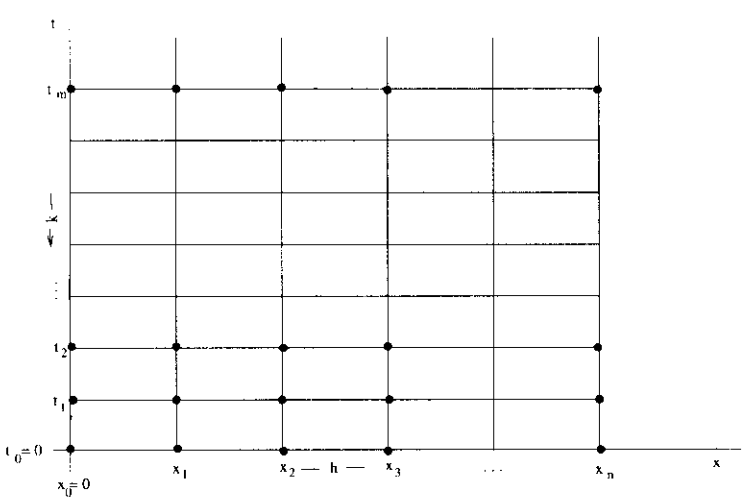


FIGURE 12.2
The region R and the mesh points.

Two formulas from Section 8.1 are used in this context: The central-difference formula for approximating $u_{xx}(x_i, t_j) = \frac{\partial^2 u}{\partial x^2}(x_i, t_j)$

$$u_{xx}(x_i, t_j) = \frac{u(x_{i+1}, t_j) - 2u(x_i, t_j) + u(x_{i-1}, t_j)}{h^2} + O(h^2) \tag{12.8}$$

and the forward-difference formula for approximating

$$\frac{\partial u}{\partial t}(x_i, t_j) = u_t(x_i, t_j) = \frac{u(x_i, t_{j+1}) - u(x_i, t_j)}{k} + O(k). \tag{12.9}$$

By substituting Eqns. (12.8) and (12.9) into Eqn. (12.5) and neglecting error terms $O(h^2)$ and $O(k)$, we obtain

$$\frac{u_i^{j+1} - u_i^j}{k} = \alpha \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2}. \tag{12.10}$$

If we set $\lambda = \alpha\,(k/h^2)$ and solve for u_i^{j+1} in Eqn. (12.10), we obtain the explicit difference formula

$$u_i^{j+1} = (1 - 2\lambda)\, u_i^j + \lambda \left(u_{i+1}^j + u_{i-1}^j \right), \tag{12.11}$$

for $i = 1, ..., n - 1$, and $j = 0, ..., m - 1$, known as the **Forward-Difference** or **classical explicit method**.

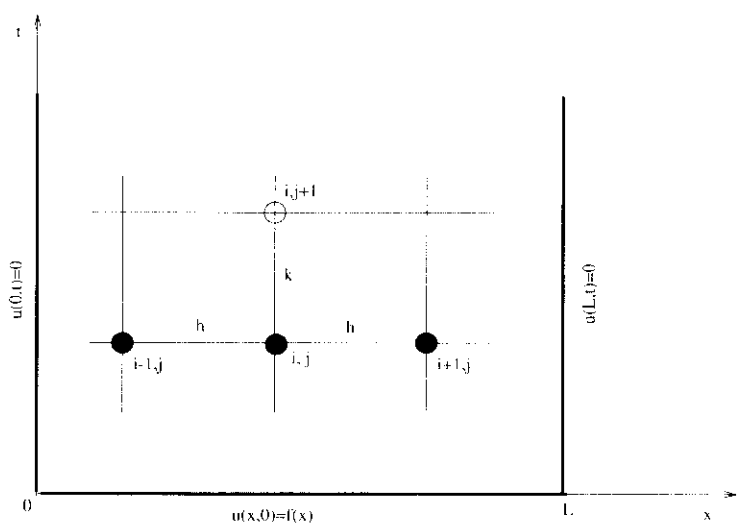


FIGURE 12.3
Schematic form of the Forward-Difference method.

In schematic form, Eqn. (12.11) is shown in Figure 12.3. The solution at every point $(i, j+1)$ on the $(j+1)$th time level is expressed in terms of the solution values at the points $(i-1, j)$, (i, j), and $(i+1, j)$ of the previous time level. Such a method is called an **explicit** method. It can be shown that the Forward-Difference method has an accuracy of the order $O(k + h^2)$ (see [21]).

The values of the initial condition $u(x_i, 0) = f(x_i)$, for $i = 0, ..., n$, are used in Eqn. (12.11) to find the values of u_i^1, for $i = 1, ..., n-1$. The boundary conditions, $u(0, t_j) = u(L, t_j) = 0$, imply that $u_0^j = u_n^j = 0$, for $j = 0, ..., m$. Once the approximations u_i^1, $i = 1, \ldots, n-1$ are known, the values of $u_i^2, u_i^3, ..., u_i^m$ can be obtained in a similar manner.

EXAMPLE 12.1

We illustrate the forward-difference method by solving the heat equation

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \quad 0 < x < 1, \quad t > 0$$

subject to the initial and boundary conditions

$$u(x, 0) = \sin \pi x, \quad 0 \le x \le 1 \quad \text{and} \quad u(0, t) = u(1, t) = 0, \quad t \ge 0$$

The exact solution to this problem is

$$u(x, t) = e^{-\pi^2 t} \sin \pi x.$$

The solution will be approximated first with $T = 0.025$, $h = 0.1$, and $k = 0.0025$ so that $\lambda = k/h^2 = 0.25$, and then with $T = 0.05$, $h = 0.1$, and $k = 0.01$ so that $\lambda = k/h^2 = 1.0$.

With $h = 0.1$ and $k = 0.0025$, the approximations of u at $t = 0.0025$ for $i = 1, 2$ are

$$\begin{aligned} u_1^1 &= \frac{1}{4}\left(u_0^0 + 2u_1^0 + u_2^0\right) = \frac{1}{4}\left[\sin(0.0) + 2\sin(\pi 0.1) + \sin(\pi 0.2)\right] \\ &= 0.301\,455, \\ u_2^1 &= \frac{1}{4}\left(u_1^0 + 2u_2^0 + u_3^0\right) = \frac{1}{4}\left[\sin(\pi 0.1) + 2\sin(\pi 0.2) + \sin(\pi 0.3)\right] \\ &= 0.573\,401. \end{aligned}$$

m-function 12.1a

The following MATLAB function **heat.m** finds the solution of the heat equation using the Forward-Difference method. INPUTS are a function *f*; the boundary condition *c1, c2*; the endpoint *L*; the maximum time *T*; the step sizes *h* and *k*; the constant α. The input function *f(x)* should be defined as an M-file.

```
function heat(f,c1,c2,L,T,h,k,alpha)
% Solve the heat equation with I.C. u(x,0)=f(x) and B.C. u(0,t)=c1
%  and u(L,t)=c2  using the Forward-Difference method.
n=L/h; m=T/k;
lambda=alpha^2*k/(h^2)
z=0:h:L;
disp('_____________________________________________________')
fprintf('      t        x =                                 ')
fprintf('%4.2f        ',z)
fprintf('\n')
disp('_____________________________________________________')
fprintf('% 5.4f  ',0)
for i=1:n+1
   u(i)=feval(f,(i-1)*h);
fprintf('%10.6f   ',u(i))
end
fprintf('\n')
for j=1:m
  t=j*k;
 fprintf('% 5.4f  ',t)
 for i=1:n+1
   if (i==1)
     y(i)=c1;
   elseif (i==n+1)
     y(i)=c2;
   else
     y(i)=(1-2*lambda)*u(i)+lambda*(u(i+1)+u(i-1));
   end;
 fprintf('%10.6f   ',y(i))
 end;
 fprintf('\n')
 u=y;
end
```

With $h = 0.1$ and $k = 0.01$, the approximations of u at $t = 0.01$ for $i = 1, 2$ are

$$u_1^1 = u_0^0 - u_1^0 + u_2^0 = \sin(0.0) - \sin(\pi 0.1) + \sin(\pi 0.2) = 0.278\,768,$$
$$u_2^1 = u_1^0 - u_2^0 + u_3^0 = \sin(\pi 0.1) - \sin(\pi 0.2) + \sin(\pi 0.3) = 0.530\,249.$$

If we continue in this manner, we get the results shown in Tables 12.1 and 12.2 obtained from using the MATLAB function heat. A three-dimensional representation of Table 12.1 is shown in Figure 12.4.

```
» heat('f1',0,0,1,.025,.1,.0025,1)

lambda =
        0.25
```

t \ x	0	0.1	0.2	0.3	0.4	0.5 ...	0.9	1
0.0000	0.000000	0.309017	0.587785	0.809017	0.951057	1.000000	0.309017	0.000000
0.0025	0.000000	0.301455	0.573401	0.789219	0.927783	0.975528	0.301455	0.000000
0.0050	0.000000	0.294078	0.559369	0.769905	0.905078	0.951655	0.294078	0.000000
0.0075	0.000000	0.286881	0.545680	0.751064	0.882929	0.928367	0.286881	0.000000
0.0100	0.000000	0.279861	0.532327	0.732685	0.861322	0.905648	0.279861	0.000000
0.0125	0.000000	0.273012	0.519300	0.714755	0.840244	0.883485	0.273012	0.000000
0.0150	0.000000	0.266331	0.506591	0.697263	0.819682	0.861865	0.266331	0.000000
0.0175	0.000000	0.259813	0.494194	0.680200	0.799623	0.840773	0.259813	0.000000
0.0200	0.000000	0.253455	0.482100	0.663554	0.780055	0.820198	0.253455	0.000000
0.0225	0.000000	0.247253	0.470303	0.647316	0.760966	0.800127	0.247253	0.000000
0.0250	0.000000	0.241202	0.458793	0.631475	0.742343	0.780546	0.241202	0.000000
Error at t = 0.025	0	0.0002465	0.0004689	0.0006453	0.0007586	0.0007977	0.000247	0

Table 12.1 Forward-Difference method for Example 12.1 with $\lambda = 0.25$.

This explicit method, just described, does not necessarily produce good results. One can see from Table 12.2 that the results obtained for the case when $\lambda = 1.0$ are poor approximations of the solution and do not reflect an accuracy of the order $O(k + h^2)$. This is due to the stability condition of the explicit method. Appropriate choices have to be made for the step sizes, h and k, that determine the values of λ. It can be shown (see [21], for further discussion) that the explicit method is stable if the *mesh ratio* λ satisfies

$$0 < \lambda \leq \frac{1}{2}. \tag{12.12}$$

This means that the error made at one stage of calculation does not cause increasingly large errors as the computations are continued, but rather will eventually damp out.

```
> heat(TF,0.0,1,.05,.1,.01,1)

lambda =
        1
```

t \ x	0	0.1	0.2	0.3	0.4	0.5 ...	0.9	1
0.00	0.000000	0.309017	0.587785	0.809017	0.951057	1.000000	0.309017	0.000000
0.01	0.000000	0.278768	0.530249	0.729825	0.857960	0.902113	0.278768	0.000000
0.02	0.000000	0.251480	0.478344	0.658384	0.773977	0.813808	0.251480	0.000000
0.03	0.000000	0.226864	0.431521	0.593937	0.698215	0.734147	0.226864	0.000000
0.04	0.000000	0.204657	0.389280	0.535798	0.629869	0.662283	0.204657	0.000000
0.05	0.000000	0.184624	0.351175	0.483351	0.568213	0.597454	0.184624	0.000000
Error at t = 0.05	0	0.004	0.0077	0.0106	0.0124	0.013	0.004	0

Table 12.2 Forward-Difference method for Example 12.1 with $\lambda = 1.0$.

12.1.2 Implicit methods

In the explicit method previously described, the approximate solution u_i^{j+1} depends on the values $u_{i-1}^j, u_i^j, u_{i+1}^j$ of u at the previous time level. Furthermore, the requirement $0 < \lambda \leq \frac{1}{2}$ places an undesirable restriction on the time step k that can be used.

The implicit method, now to be described, overcomes the stability requirement by being unconditionally stable. The finite difference equation of this method is obtained by replacing $u_{xx}(x, y)$ in Eqn. (12.5) with the average of the centered-difference at the time steps $j+1$ and j and $u_t(x, y)$ with the forward-difference. The result is

$$\frac{u_i^{j+1} - u_i^j}{k} = \frac{\alpha}{2}\left[\frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{h^2} + \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{h^2}\right]. \quad (12.13)$$

By setting $\lambda = \alpha k/h^2$ as before, Eqn. (12.13) can be written as

$$-\lambda u_{i-1}^{j+1} + 2(1+\lambda)\, u_i^{j+1} - \lambda u_{i+1}^{j+1} = \lambda u_{i-1}^j + 2(1-\lambda)\, u_i^j + \lambda u_{i+1}^j \quad (12.14)$$

for $i = 1, 2, ..., n-1$. This method is called the **Crank-Nicolson** method. In schematic form, Eqn. (12.14) is shown in Figure 12.5. The solution value at any point $(i, j+1)$ on the $(j+1)$th time level is dependent on the solution values at the neighboring points on the same level and three points on the jth time level. Since values at the $(j+1)$th time level are obtained implicitly, the method is called an **implicit** method. It can be shown that the Crank-Nicolson method has an accuracy of the order $O(k^2 + h^2)$ and is unconditionally stable

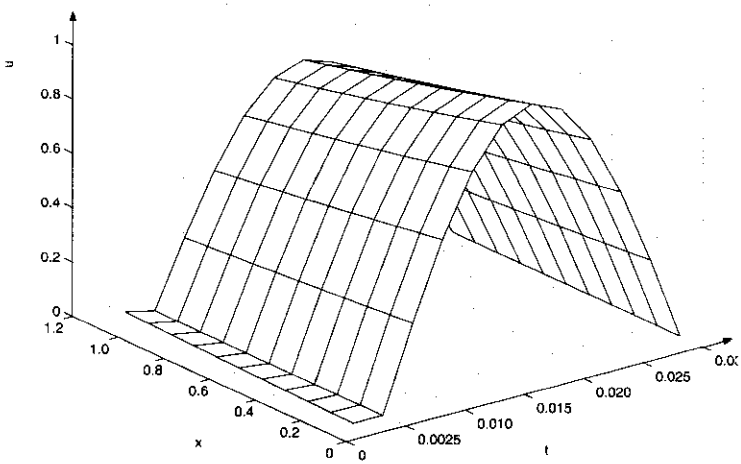


FIGURE 12.4
$u = u_{ij}$ **for Example 12.1 with** $\lambda = 0.25$**.**

(see [21]).

If, in Eqn. (12.5) we replace the time derivative by a forward-difference and the space derivative by a centered difference at the forward time step $j+1$, we get the **classical implicit** method

$$u_i^j = -\lambda u_{i-1}^{j+1} + (1+2\lambda)u_i^{j+1} - \lambda u_{i+1}^{j+1}$$

where $\lambda = \alpha k/h^2$. This method is unconditionally stable.

The matrix form of the Crank-Nicolson method is

$$A\,\mathbf{u}^{(j+1)} = B\,\mathbf{u}^{(j)}, \quad j = 0, 1, 2, \ldots \tag{12.15}$$

where $\mathbf{u}^{(j+1)} = \left[u_1^{j+1}, u_2^{j+1}, \ldots, u_{n-1}^{j+1}\right]^t$,

$$A = \begin{bmatrix} 2(1+\lambda) & -\lambda & 0 & \ldots & 0 \\ -\lambda & 2(1+\lambda) & -\lambda & \ddots & \vdots \\ 0 & -\lambda & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\lambda \\ 0 & \ldots & 0 & -\lambda & 2(1+\lambda) \end{bmatrix},$$

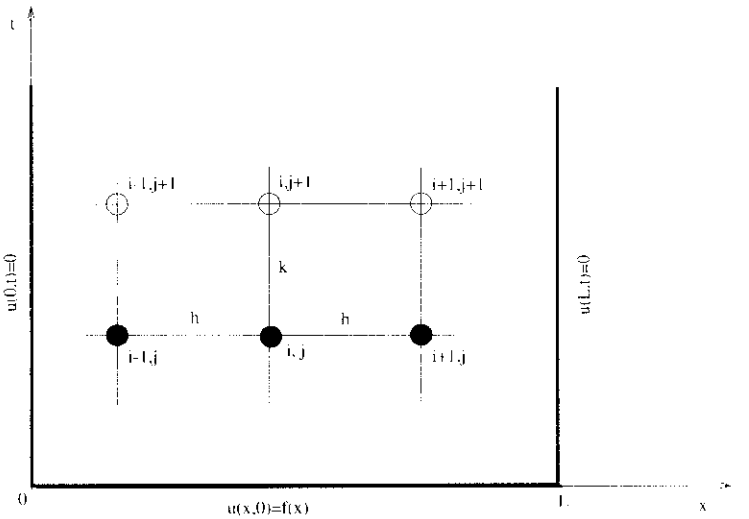


FIGURE 12.5
Schematic form of the Crank-Nicolson method.

and

$$B = \begin{bmatrix} 2(1-\lambda) & \lambda & 0 & \dots & 0 \\ \lambda & 2(1-\lambda) & \lambda & \ddots & \vdots \\ 0 & \lambda & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \lambda \\ 0 & \dots & 0 & \lambda & 2(1-\lambda) \end{bmatrix}.$$

The tridiagonal matrix A is positive definite and strictly diagonally dominant. Therefore, A is nonsingular and the system of equation (12.14) can be solved by any method described in Chapter 4.

EXAMPLE 12.2

Solve the heat equation

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \quad 0 < x < 1, \quad 0 < t < 0.5$$

subject to the initial and boundary conditions

$$u(x,0) = \sin \pi x, \quad 0 \le x \le 1$$
$$u(0,t) = u(1,t) = 0, \quad 0 \le t < 0.5$$

using the Crank-Nicolson method.

m-function 12.1b

The following MATLAB function **heat_crank.m** finds the solution of the heat equation using the Crank-Nicolson method. INPUTS are a function *f*; the boundary condition *c1, c2*; the endpoint *L*; the maximum time *T*; the step sizes *h* and *k*; the constant α. The input function *f(x)* should be defined as an m-file.

```
function heat_crank(f,c1,c2,L,T,h,k,alpha)
% Solve the heat equation with I.C. u(x,0)=f(x) and B.C. u(0,t)=c1,
% and u(L,t)=c2 using Crank-Nicolson method.
n=L/h; m=T/k; lambda=alpha^2*k/(h^2)
z=0:h:L;
disp('_____________________________________________________')
fprintf('          t          x = ')
fprintf('%4.2f        ',z)
fprintf('\n')
disp('_____________________________________________________')
fprintf('% 4.2f  ',0)
for i=1:n+1
  u(i)=feval(f,(i-1)*h);
fprintf('%10.6f   ',u(i))
end
fprintf('\n')
for i=2:n
  if (i~=n)
     a(i)=-lambda;
  end
  b(i)=2*(1+lambda);
  if (i~=n)
     c(i)=-lambda;
  end
end
bb=b;
for j=1:m
 t=j*k;
 fprintf('% 4.2f  ',t)
 for i=2:n
   d(i)=lambda*u(i-1)+2*(1-lambda)*u(i)+lambda*u(i+1);
 end
 y(n+1)=c1;
 y(1)=c2;
 for i=3:n
   ymult=a(i-1)/bb(i-1);
```

```
    bb(i)=bb(i)-ymult*c(i-1);
    d(i)=d(i)-ymult*d(i-1);
  end
  y(n)=d(n)/bb(n);
  for i=n-1:-1:2
    y(i)=(d(i)-c(i)*y(i+1))/bb(i);
  end
    for i=1:n+1
      fprintf('%10.6f  ',y(i))
    end
  fprintf('\n')
  u=y;
  bb=b;
end
```

We choose $h = 0.2$, $k = 0.05$, so that $\lambda = 1.25$, $n = 5$, and $m = 10$. By replacing the value of λ into (12.14), we get the simplified difference equation

$$-1.25\,u_{i-1}^{j+1} + 4.5u_i^{j+1} - 1.25\,u_{i+1}^{j+1} = 1.25\,u_{i-1}^{j} - 0.5\,u_i^{j} + 1.25\,u_{i+1}^{j}$$

for $i = 1, ..., 4$. At the first time step $t = k$, u_i^1 is given by the solution of the tridiagonal system

$$\begin{bmatrix} 4.5 & -1.25 & 0 & 0 \\ -1.25 & 4.5 & -1.25 & 0 \\ 0 & -1.25 & 4.5 & -1.25 \\ 0 & 0 & -1.25 & 4.5 \end{bmatrix} \begin{bmatrix} u_1^1 \\ u_2^1 \\ u_3^1 \\ u_4^1 \end{bmatrix} = \begin{bmatrix} -0.5\,u_1^0 + 1.25\,u_2^0 \\ 1.25\,u_1^0 - 0.5\,u_2^0 + 1.25\,u_3^0 \\ 1.25\,u_2^0 - 0.5\,u_3^0 + 1.25\,u_4^0 \\ 1.25\,u_3^0 - 0.5\,u_4^0 \end{bmatrix}$$

$$= \begin{bmatrix} 0.894\,928 \\ 1.448\,024 \\ 1.448\,024 \\ 0.894\,928 \end{bmatrix}$$

where $u_i^0 = \sin(\pi i h)$. The solution to the tridiagonal system is

$$u(x_i, 0.05) = [0.361\,228\,40, 0.584\,479\,83, 0.584\,479\,83, 0.361\,228\,40]^t$$

for $i = 1, 2, 3, 4$. Using the MATLAB function heat_crank, gives the solution shown in Table 12.3.

```
» heat_crank('f1',0,0,1,0.5,0.2,0.05,1)

lambda =
          1.25
```

t \ x	0	0.2	0.4	0.6	0.8	1
0.00	0.000000	0.587785	0.951057	0.951057	0.587785	0.000000
0.05	0.000000	0.361228	0.584480	0.584480	0.361228	0.000000
0.10	0.000000	0.221996	0.359197	0.359197	0.221996	0.000000
0.15	0.000000	0.136430	0.220748	0.220748	0.136430	0.000000
0.20	0.000000	0.083844	0.135662	0.135662	0.083844	0.000000
0.25	0.000000	0.051527	0.083372	0.083372	0.051527	0.000000
0.30	0.000000	0.031666	0.051237	0.051237	0.031666	0.000000
0.35	0.000000	0.019461	0.031488	0.031488	0.019461	0.000000
0.40	0.000000	0.011960	0.019351	0.019351	0.011960	0.000000
0.45	0.000000	0.007350	0.011893	0.011893	0.007350	0.000000
0.50	0.000000	0.004517	0.007309	0.007309	0.004517	0.000000
Error at t = 0.5	0	0.00029	0.000469	0.000469	0.00029	0

Table 12.3 Crank-Nicolson method for Example 12.2 with $\lambda = 1.25$.

EXERCISE SET 12.1

1. Verify that $u(x,t) = e^{-t}\cos\pi(x - \frac{1}{2})$ is a solution to the heat equation $\frac{1}{\pi^2}u_{xx} = u_t$.

2. In which part of the xy-plane is the following equation parabolic?

$$u_{xx} + 4u_{xy} + (x^2 + y^2)u_{yy} = \sin(xy)$$

3. Approximate the solution of the following parabolic partial-differential equations using the Forward-Difference:

   (a) $u_{xx} = u_t, \quad 0 < x < 1,\ 0 < t < 0.1;$

   $u(0,t) = u(1,t) = 0, \quad 0 \le t \le 0.1;$

   $$u(x,0) = \begin{cases} 1, & 0 \le x < 0.5, \\ 0, & 0.5 < x \le 1. \end{cases}$$

   Use $h = 0.2$, $k = 0.01$.

   (b) $u_{xx} = u_t, \quad 0 < x < 1,\ 0 < t < 0.2;$

$u(0,t) = u(1,t) = 0, \quad 0 \le t \le 0.2;$

$u(x,0) = 4x - 4x^2, \quad 0 \le x \le 1.$

Use $h = 0.2$, $k = 0.02$.

(c) $10\,u_{xx} = u_t, \quad 0 < x < 100,\ 0 < t < 0.05;$

$u(0,t) = u(100,t) = 0, \quad 0 \le t \le 0.05;$

$$u(x,0) = \begin{cases} 0 & 0 \le x < 25, \\ 50 & 25 \le x \le 75, \\ 0 & 75 < x \le 100. \end{cases}$$

Use $h = 10$, $k = 0.01$.

4. Use the Crank-Nicolson method to approximate the solution to the following parabolic partial-differential equations:

(a) $3\,u_{xx} = u_t, \quad 0 < x < \pi,\ 0 < t < 0.1;$

$u(0,t) = u(\pi,t) = 0, \quad 0 \le t \le 0.1;$

$u(x,0) = \sin x, \quad 0 \le x \le \pi.$

Use $h = 0.52$, $k = 0.01$.

(b) $\frac{1}{\pi^2}\,u_{xx} = u_t, \quad 0 < x < 1,\ 0 < t < 0.1;$

$u(0,t) = u(1,t) = 0, \quad 0 \le t \le 0.1;$

$u(x,0) = \sin \pi x, \quad 0 \le x \le 1.$

Use $h = 0.1$, $k = 0.01$. Compare your results with the exact solution $u(x,t) = e^{-t}\sin \pi x$.

(c) $u_{xx} = u_t, \quad 0 < x < 1,\ 0 < t < 0.05;$

$u(0,t) = u(\pi,t) = 0, \quad 0 \le t \le 0.05;$

$u(x,0) = \sin \pi x + \sin 2\pi x, \quad 0 \le x \le 1.$

Use $h = 0.2$, $k = 0.01$.

5. Derive a finite difference method for solving the nonlinear parabolic equation

m-function 12.2

The following MATLAB function **hyperbolic.m** finds the solution of the hyperbolic equation using a three level explicit method. INPUTS are functions *f* and *g*; the boundary condition *c1, c2*; the endpoint *L*; the maximum time *T*; the step sizes *h* and *k*; the constant α. The input functions *f(x)* and *g(x)* should be defined as M-files.

```
function hyperbolic(f,g,c1,c2,L,T,h,k,alpha)
% Solve the hyperbolic equation with u(x,0)=f(x), u_t(x,0)=g(x) and
% u(0,t)=c1, u(L,t)=c2 using a three level explicit method.
n=L/h; m=T/k;
lambda=alpha*k/h;
z=0:h:L;
disp('_____________________________________________')
fprintf('          t              x =                    ')
fprintf('%4.2f       ',z)
fprintf('\n')
disp('_____________________________________________')
fprintf('% 4.2f  ',0)
for i=1:n+1
  u0(i)=feval(f,(i-1)*h);
  fprintf('%10.6f   ',u0(i))
end
fprintf('\n')
fprintf('% 4.2f  %10.6f   ',k,c1)
for i=1:n-1
  u1(i+1)=(1-lambda^2)*feval(f,i*h)+lambda^2/2*(feval(f,(i+1)*h)...
      +feval(f,(i-1)*h))+k*feval(g,i*h);
  fprintf('%10.6f   ',u1(i+1))
end
fprintf('%10.6f   ',c2)
fprintf('\n')
for j=2:m
 t=j*k;
 fprintf('% 4.2f  ',t)
 u1(1)=c1; ui(1)=c1; u1(n+1)=c2; ui(n+1)=c2;
 fprintf('%10.6f   ',ui(1))
 for i=2:n
   ui(i)=2*(1-lambda^2)*u1(i)+lambda^2*(u1(i+1)+u1(i-1))-u0(i);
   fprintf('%10.6f   ',ui(i))
 end
 fprintf('%10.6f   ',ui(n+1))
```

```
  fprintf('\n')
  u0=u1;
  u1=ui;
end
```

For $t = 2k = 0.1$, we get the difference equation

$$u_i^2 = u_{i+1}^1 + u_{i-1}^1 - u_i^0, \quad i = 1, 2, 3, 4$$

which implies that

$$\begin{aligned}
i = 1,\ u_1^2 &= u_2^1 - u_1^0 = 0.181\,635\,63 \\
i = 2,\ u_2^2 &= u_3^1 + u_1^1 - u_2^0 = 0.29389263 \\
i = 3,\ u_2^2 &= u_4^1 + u_2^1 - u_3^0 = 0.29389263 \\
i = 4,\ u_3^2 &= u_3^1 - u_4^0 = 0.181\,635\,63
\end{aligned}$$

Using the MATLAB function hyperbolic with $T = 0.5$, $h = 0.2$, and $k = 0.05$ gives the solution shown in Table 12.4. A three-dimensional representation of Table 12.4 is shown in Figure 12.8.

» hyperbolic('f1','g1',0,0,1,0.5,0.2,0.05,4)

lambda =
1.0

t \ x	0	0.2	0.4	0.6	0.8	1.0
0.00	0.000000	0.587785	0.951057	0.951057	0.587785	0.000000
0.05	0.000000	0.475528	0.769421	0.769421	0.475528	0.000000
0.10	0.000000	0.181636	0.293893	0.293893	0.181636	0.000000
0.15	0.000000	-0.181636	-0.293893	-0.293893	-0.181636	0.000000
0.20	0.000000	-0.475528	-0.769421	-0.769421	-0.475528	0.000000
0.25	0.000000	-0.587785	-0.951057	-0.951057	-0.587785	0.000000
0.30	0.000000	-0.475528	-0.769421	-0.769421	-0.475528	0.000000
0.35	0.000000	-0.181636	-0.293893	-0.293893	-0.181636	0.000000
0.40	0.000000	0.181636	0.293893	0.293893	0.181636	0.000000
0.45	0.000000	0.475528	0.769421	0.769421	0.475528	0.000000
0.50	0.000000	0.587785	0.951057	0.951057	0.587785	0.000000
Error at t = 0.5	0	0	1.1e-16	1.1e-16	1.1e-16	0

Table 12.4 Explicit three level difference method for Example 12.3.

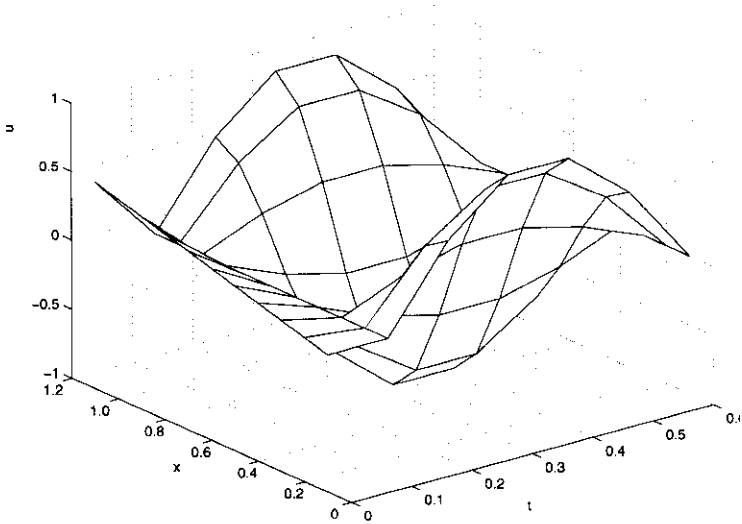


FIGURE 12.8
Explicit three level difference method for Example 12.3.

EXERCISE SET 12.2

1. In which part of the xy-plane is the following equation hyperbolic?

$$u_{xx} + 4u_{xy} + (x^2 + y^2)u_{yy} = \sin(xy)$$

2. Approximate the solution of the following hyperbolic partial-differential equations using the three level difference method

   (a) $u_{xx} = u_{tt}, \quad 0 < x < 1, 0 < t < 0.5;$

   $u(0,t) = u(1,t) = 0, \quad 0 \le t \le 0.1;$

   $u_t(x,0) = x^2, \; u(x,0) = x, \quad 0 \le x \le 1.$
   Use $h = 0.1$, $k = 0.05$.

   (b) $u_{xx} = u_{tt}, \quad 0 < x < \pi, 0 < t < 1.0;$

   $u(0,t) = u(\pi,t) = 0, \quad 0 \le t \le 1.0;$

   $u_t(x,0) = 0, \quad u(x,0) = 2x/\pi(\pi - x), \quad 0 \le x \le \pi.$

   Use $h = \pi/5$, $k = 0.1$.

(c) $u_{xx} = u_{tt}, \quad 0 < x < 1, 0 < t < 1;$

$u(0,t) = u(1,t) = 0, \quad 0 \le t \le 1;$

$u_t(x,0) = \sin \pi x, \quad u(x,0) = \sin 2\pi x, \quad 0 \le x \le 1.$

Use $h = 0.25$, $k = 0.25$.

(d) $u_{xx} = u_{tt}, \quad 0 < x < \pi, 0 < t < 1.0;$

$u(0,t) = u(\pi,t) = 0, \quad 0 \le t \le 1.0;$

$u_t(x,0) = \sin x, \quad u(x,0) = \sin x, \quad 0 \le x \le \pi.$

Use $h = \pi/10$, $k = 0.25$.

(e) $u_{xx} = u_{tt}, \quad 0 < x < 4, t > 0;$

$u(0,t) = u(4,t) = 0, \quad 0 \le t \le 1.0;$

$$u(x,0) = \begin{cases} 16x^2(1-x)^2, & 0 \le x \le 1, \\ 0, & 1 \le x \le 4. \end{cases}$$

$u_t(x,0) = 0, \quad 0 \le x \le 4.$

Use $h = 0.4$, $k = 0.25$.

12.3 ELLIPTIC EQUATIONS

m-function 12.3

The following MATLAB function **laplace.m** finds the solution of the Laplace's equation using a five-point difference formula. INPUTS are functions *f1, f2, g1*, and *g2*; the endpoint *a*; the number of subintervals *n*; the maximum number of iterations *itmax*, and a tolerance *tol*. The input functions should be defined as m-files.

```
function laplace(f1,f2,g1,g2,a,n,itmax,tol)
% Solve the laplace equation in a square with B.C.  u(x,0)=f1(x)
% u(x,a)=f2(x)  and u(0,y)=g1(y) and u(a,y)=g2(y) using a
% five-point difference method.
h=a/n;
z=0:h:a;
h
disp('_____________________________________________________')
fprintf('          u=  x\\y                                  ')
fprintf('%4.2f      ',z)
fprintf('\n')
disp('_____________________________________________________')
for i=1:n+1
  u(i,1)=feval(f1,(i-1)*h);
  u(i,n+1)=feval(f2,(i-1)*h);
  u(1,i)=feval(g1,(i-1)*h);
  u(n+1,i)=feval(g2,(i-1)*h);
end
iter=0;
err=tol+1;
while (err>tol)&(iter<itmax)
   err=0;
  for i=2:n
    for j=2:n
      oldu=u(i,j);
      u(i,j)=(u(i+1,j)+u(i-1,j)+u(i,j+1)+u(i,j-1))/4;
      res=abs(u(i,j)-oldu);
      if (err<res);
        err=res;
      end
    end
  end
  iter=iter+1;
end
for i=1:n+1
  fprintf('   %4.2f',z(i))
```

```
    for j=1:n+1
      fprintf('%10.4f  ',u(i,j))
    end
    fprintf('\n')
  end
  iter
```

```
laplace('f1','f2','g1','g2', pi,9,120,10^(-6))

h =
                0.3489
```

u= x\y	0.35	0.70	1.05	1.40	1.75 ...	2.79	3.14
0.00	0	0	0	0	0	0	0
0.35	0.0108	0.0228	0.0377	0.0570	0.0833	0.2416	0.3420
0.70	0.0202	0.0429	0.0708	0.1072	0.1566	0.4541	0.6428
1.05	0.0273	0.0578	0.0954	0.1444	0.2109	0.6118	0.8660
1.40	0.0310	0.0658	0.1085	0.1643	0.2399	0.6957	0.9848
1.75	0.0310	0.0658	0.1085	0.1643	0.2399	0.6957	0.9848
2.09	0.0273	0.0578	0.0954	0.1444	0.2109	0.6118	0.8660
2.44	0.0202	0.0429	0.0708	0.1072	0.1566	0.4541	0.6428
2.79	0.0108	0.0228	0.0377	0.0570	0.0833	0.2416	0.3420
3.14	0	0	0	0	0	0	0

Table 12.5 Approximate solution for Example 12.5.

EXERCISE SET 12.3

1. Solve Laplace's equation for a square $0 < x < a$ and $0 < y < a$, subject to the specified boundary conditions.

   (a) $u(x,0) = u(0,y) = u(x,2) = 0$, $u(2,y) = 10$.

   (b) $u(x,0) = u(0,y) = u(x,\pi) = 0$, $u(\pi,y) = \sin y$.

   (c) $u(x,0) = u(x,\pi) = u(\pi,y) = 0$, $u(0,y) = \sin^2 y$.

   (d) $u(x,0) = u(0,y) = u(x,1) = 0$, $u(1,y) = -y(y-1)$.

APPLIED PROBLEMS FOR CHAPTER 12

1. Consider a homogeneous iron rod 100 cm long that is heated to the initial temperature distribution $f(x) = 100 - x$, $0 \leq x \leq 100$. At time $t = 0$ the lateral surface is insulated, and the left end of the rod is plunged into a reservoir of oil kept at 30^oC, while the right end remains at the constant temperature of 70^oC. The thermal diffusivity α of iron is 0.15 cm^2/sec. Use $h = 10$ in. and an appropriate value of k so that $\lambda = \alpha 2k/h^2 = 0.5$. Carry on the solution for five time steps.

2. A large steel plate 2 cm thick is heated according to the initial temperature distribution $f(x) = x$ if $0 \leq x \leq 1$ and $f(x) = 2 - x$ if $1 \leq x \leq 2$. Both faces are maintained at 0^oC. The thermal diffusivity α of the steel is 0.15 cm^2/sec. Assume that the lateral flow of heat relative to the flow perpendicular to the faces is neglected. Use the heat equation with $h = 0.25$ in. and an appropriate value of k so that $\lambda = \alpha 2k/h^2 = 0.5$ to carry on the solution for five time steps.

3. Solve the wave equation

Appendix A

Calculus Review

We assume that the reader is familiar with topics normally covered in the undergraduate calculus sequence. For purpose of reference this appendix gives a summary of the relevant results from calculus on limits, continuity, differentiation, and integration. The proof of these results can be found in almost any calculus book.

A.0.1 Limits and continuity

DEFINITION A.1 *Let a function f be defined in an open interval and L be a real number. We write*

$$\lim_{x \to a} f(x) = L$$

if for every $\epsilon > 0$, there is a $\delta > 0$ such that

$$\text{if} \quad 0 < |x - a| < \delta, \qquad \text{then} \quad 0 < |f(x) - L| < \epsilon.$$

DEFINITION A.2 *A function f is continuous at $x = a$ if it satisfies the following three conditions:*

(a) *f is defined at $x = a$.*

(b) $\lim_{x \to a} f(x) =$ *exists.*

(c) $\lim_{x \to a} f(x) = f(a)$.

THEOREM A.1
A polynomial function f is continuous at each point of the real line.

THEOREM A.2
Let f be a continuous function on the interval $[a, b]$. Then $f(x)$ assumes its maximum and minimum values on $[a, b]$; that is, there are numbers $x_1, x_2 \in [a, b]$ such that

$$f(x_1) \leq f(x) \leq f(x_2)$$

for all $x \in [a, b]$.

A.0.2 Differentiation

DEFINITION A.3 *The derivative of a function f is the function f' defined by*

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

provided the limit exists.

For example, let $f(x) = \sqrt{x}$, then if $x > 0$

$$\begin{aligned} f'(x) &= \lim_{h \to 0} \frac{\sqrt{x+h} - \sqrt{x}}{h} \\ &= \lim_{h \to 0} \frac{\sqrt{x+h} - \sqrt{x}}{h} \cdot \frac{\sqrt{x+h} + \sqrt{x}}{\sqrt{x+h} + \sqrt{x}} \\ &= \lim_{h \to 0} \frac{1}{\sqrt{x+h} + \sqrt{x}} = \frac{1}{2\sqrt{x}}. \end{aligned}$$

THEOREM A.3
A polynomial function f is differentiable at each point of the real line.

THEOREM A.4
If a function f is differentiable at $x = a$, then f is continuous at a.

A.0.3 Integration

THEOREM A.5 (Fundamental Theorem for Calculus)
Suppose that f is continuous on the closed interval $[a, b]$.

Part I: If the function F is defined on $[a, b]$ by

$$F(x) = \int_a^x f(t)\, dt,$$

then F is an antiderivative of f for x in $[a, b]$.

Part II: If G is any antiderivative of f on $[a, b]$, then

$$\int_a^b f(x)\,dx = G(b) - G(a).$$

For example,

$$\int_1^4 \sqrt{x}\,dx = \left[\frac{2}{3}\,x^{3/2}\right]_1^4 = \frac{2}{3}\,(4)^{3/2} - \frac{2}{3}\,(1)^{3/2} = \frac{14}{3}$$

THEOREM A.6 (First Mean Value Theorem for Integrals)
If f is continuous on $[a, b]$, then

$$\int_a^b f(x)\,dx = f(\xi)(b-a)$$

for at least one number ξ in the open interval (a, b).

For example,

$$\int_0^3 x^2\,dx = 9 = f(\xi)(3-0) = 3\,\xi^2.$$

This implies that $\xi = \sqrt{3} \in (0, 3)$ satisfies the conclusion of the theorem.

THEOREM A.7 (Second Mean Value Theorem for Integrals)
If f is continuous on $[a, b]$ and if the function $g(x)$ is of constant sign in the interval (a, b), then

$$\int_a^b f(x)\,g(x)\,dx = f(\xi)\int_a^b g(x)\,dx$$

for at least one number ξ in the open interval (a, b).

THEOREM A.8
Let f be continuous on $[a, b]$ and $a \le c \le b$, then

$$\frac{d}{dx}\int_c^x f(t)\,dt = f(x)$$

for every x in $[a, b]$.

THEOREM A.9 (Integration by Parts)
Let $u(x)$ and $v(x)$ be real-valued functions with continuous derivatives. Then

$$\int u'(x)\, v(x)\, dx = u(x)v(x) - \int u(x)\, v'(x)\, dx$$

For example,

$$\int xe^x\, dx = xe^x - \int e^x\, dx = xe^x - e^x + C.$$

Appendix B

MATLAB Built-in Functions

Listed below are some of the MATLAB built-in functions grouped by subject areas.

Special Matrices

diag	diagonal
eye	identity
hadamard	Hadamard
hankel	Hankel
hilb	Hilbert
invhilb	inverse Hilbert
linspace	linearly spaced vectors
logspace	logarithmically spaced vectors
magic	magic square
meshdom	domain for mesh points
ones	constant
pascal	Pascal
rand	random elements
toeplitz	Toeplitz
vander	Vandermonde
zeros	zero

Interpolation

spline	cubic spline
table1	1-D table look-up
table2	2-D table look-up

Numerical Integration

quad	numerical function integration
quad8	numerical function integration

Differential Equation Solution

ode23	2nd and 3rd order Runge-Kutta method
ode45	4th and 5th order Runge-Kutta-Fehlberg method

Graph Paper

plot	linear X-Y plot
loglog	loglog X-Y plot
semilogx	semi-log X-Y plot
semilogy	semi-log X-Y plot
polar	polar plot
mesh	3-dimensional mesh surface
contour	contour plot
meshdom	domain for mesh plots
bar	bar charts
stairs	stairstep graph
errorbar	add error bars

Elementary Math Functions

abs	absolute value or complex magnitude
angle	phase angle
sqrt	square root
real	real part
imag	imaginary part
conj	complex conjugate
round	round to nearest integer
fix	round toward zero
floor	round toward $-\infty$
ceil	round toward ∞
sign	signum function
rem	remainder
exp	exponential base e
log	natural logarithm
log10	log base 10

Text and Strings

abs	convert string to ASCII values
eval	evaluate text macro
num2str	convert number to string
int2str	convert integer to string
setstr	set flag indicating matrix is a string
sprintf	convert number to string
isstr	detect string variables
strcomp	compare string variables
hex2num	convert hex string to number

Control Flow

if	conditionally execute statements
elseif	used with if
else	used with if
end	terminate bif, for, while
for	repeat statements a number of times
while	do while
break	break out of for and while loops
return	return from functions
pause	pause until key pressed

Programming and M-files

input	get numbers from keyboard
keyboard	call keyboard as M-file
error	display error message
function	define function
eval	interpret text in variables
feval	evaluate function given by string
echo	enable command echoing
exist	check if variables exist
casesen	set case sensitivity
global	define global variables
startup	startup M-file
getenv	get environment string
menu	select item from menu
etime	elapsed time

Decompositions and Factorizations

backsub	back substitution
chol	Cholesky factorization
eig	eigenvalues and eigenvectors
inv	inverse
lu	factors from Gaussian elimination
nnls	non-negative least squares
orth	orthogonalization
qr	orthogonal-triangular decomposition
rref	reduced row echelon form
svd	singular value decomposition

Appendix C

Text MATLAB Functions

In this appendix we list all the MATLAB functions that are supplied in this text. These functions are contained in a CD-ROM attached to the cover of the book.

MATLAB Functions	Definition	Section
abash	Adams-Bashforth of orders 2, 3, and 4	9.6
amoulton	Adams-Moulton method	9.8
bisect	Bisection method	3.1
derive	Differentiation	8.1
euler	Euler's method	9.1
explsqr	Least squares fit, exponential	7.3
false	False position method	3.2
finitediff	Finite difference methods	10.1
fixed	Fixed point iteration	3.6
gaussel	Gaussian elimination, scaled pivoting	4.3
gauss_quad	Gaussian quadrature	8.4
heat	Parabolic differential equations	12.1
heat_crank	Parabolic differential equations	12.1
hyperbolic	Hyperbolic differential equations	12.2
hyplsqr	Least squares fit, hyperbolic	7.3
jacobi	Jacobi iteration	4.6
lagrange	Lagrange interpolation polynomial	5.4
laplace	Laplace's equation	12.3
linlsqr	Least squares fit, linear	7.1
lshoot	Linear shooting method	10.2
lufact	LU decomposition	4.4

MATLAB Functions	Definition	Section
midpoint	Midpoint method	9.4
newton	Newton's method	3.4
newton_sys	Newton's method for systems	3.8
newton2	Newton's method, multiple roots	3.7
newtondd	Interpolation, Newton differences	5.2
ngaussel	Gaussian elimination, naive	4.2
polylsqr	Least squares fit, polynomial	7.2
rk2_4	Runge-Kutta method, orders 2 and 4	9.4
romberg	Romberg integration	8.3
secant	Secant method	3.5
seidel	Gauss-Seidel iteration	4.7
simpson	Simpson's rule	8.2
spl1	Spline, linear	6.1
spl3	Spline, cubic	6.3
sys_rk4	System of differential equations	9.11
trapez	Trapezoidal rule	8.1

Answers to Selected Exercises

CHAPTER 1

Section 1.3

1. $f(-0.88) \approx 2.381$.

3. $f(0.2) \approx 0.2426667$ for $n = 3$.

7. (a) No (b) $c = 2/(3\sqrt{3})$ (c) No

8. $\{\sqrt{3}\}$.

9. No.

CHAPTER 2

Section 6.3

2. $S(1.6) = 3.04$.

3. 1.876.

4. $a = 6$, $b = -18$, $c = -36$, $d = -24$.

6. No.

7. $S(2.05) = 1.6075$, $S(2.15) = 2.84$, $S(2.25) = 4.5325$..

CHAPTER 7

Section 7.1

1. $y = 1.97x + 0.08$, $E(a, b) = 1.059$.

2. $y = 3.14x + 23$, $E(a, b) = 41.71$.

3. $y = -1.77x + 80$, $E(a, b) = 1.86E + 3$. homework #13 = 57.

Section 7.2

1. $p(x) = 0.488 + 1.65x + 0.05x^2$.

2. (a) $p(x) = 26.72 - 52.34x + 29.92x^2$, $E = 67.97$.

   (b) $p(x) = 115.26 - 211.51x + 118.64x^2 - 15.51x^3$, $E = 31.73$.

3. $p(x) = 12.59 - 9.99x + 1.95x^2$, $f(6.0) \approx 22.70$.

4. The polynomial of degree two is
   $p(x) = 7 + 3x + x^2$, E = $1.2E - 27$.

Section 7.3

1. $y = 1.99e^{.48x}$, $E = 0.002677$, $y(5.0) \approx 14.67$.

2. $y = 2.57 + 1.21/x$, $y(3.0) \approx 2.97$.

CHAPTER 8

Section 8.1

4. $f'(1) \approx 2.722814$.

Section 8.2

1. (a) 1.707107 (c) -4.209663 (e) 0.792624.

4. 0.1.003353 for $h = 0.1$; 1.00003 for $h = 0.01$.

5. $\ln 6 \approx 1.79175945$ for $n = 3$ and $h = 0.1$.

6. $\ln 3 \approx 1.09861195$ for $n = 2$ and $h = 0.1$.

Section 8.3

1. (a) 0.292290 (c) -0.433155.

2. 3.1415965.

4. 0.5381721 for $h = 0.04$, $E = 17(0.2)(0.04)^2 e^{1.25}/12 \approx 0.001505$.

6. 1.436004, $n = 5$,
   1.429270, $n = 7$,
   1.426503, $n = 9$.

Section 8.4

3. 1.5801713

4. $h \approx 0.15331$.

5. 0.000002.

6. $0.940012 < 1 < 1.061267$.

Section 8.5

1. 1.09863055, $E = 0.000018$.

3. 0.8591451666

5. 82/15

6. 418.408987.

Section 8.6

1. (a) 0.292893 (b) 1.656840 (c) -0.43989 (d) 1.238583

2. 0.786885, $n = 2$.
   0.785267, $n = 3$,
   0.785402, $n = 4$.

3. 0.627393.

4. -0.9059848, $n = 5$.

5. 5.300966.

CHAPTER 9

Section 9.1

1. (a)

t_i	$y_i(h = 0.1)$
0.2	2.56
0.4	1.6384
0.6	1.048576
0.8	0.671088
1.0	0.429496

(b)

t_i	$y_i(h = 0.1)$
1.1	0.765902
1.2	0.731161

(c)

t_i	$y_i(h = 0.1)$
0.2	2.401603
0.4	1.823684
0.6	1.295803
0.8	0.846296
1.0	0.496120

(d)

t_i	$y_i(h = 0.1)$
1.9	-0.000290
2.0	0.000514

3.

t_i	$y_i(h = 0.1)$
1.2	0.504545
1.4	0.524791
1.6	0.557778
1.8	0.602524
2.0	0.659112

$f(t) = 0.655812 - 0.31609t + 0.159036t^2$.

(i) $y(1.02) \approx f(1.02) = 0.49885892$, $E = 0.0012$.
(ii) $y(1.67) \approx f(1.67) = 0.57147362$, $E = 0.0136$.
(iii) $y(1.98) \approx f(1.98) = 0.65343451$, $E = 0.0194$.

5.

t_i	$y_i(h = 0.1)$	Error
0.2	0.01	0.00960
0.4	0.057824	0.01610
0.6	0.133878	0.01728
0.8	0.222919	0.01343
1.0	0.309146	0.00691

6.

(a)

t_i	$y_i(h = 0.1)$
0.2	1.98
0.4	1.882188
0.6	1.716555
0.8	1.500612
1.0	1.256313

(b)

t_i	$y_i(h = 0.2)$
0.2	1.0
0.4	0.992053
0.6	0.961147
0.8	0.896023
1.0	0.793180

(c)

t_i	$y_i(h = 0.2)$
1.2	1.2
1.4	1.4
1.6	1.6
1.8	1.8
2.0	2.0

(e)

t_i	$y_i(h = 0.1)$
2.2	3.332497
2.4	3.662115
2.6	3.989580
2.8	4.315387
3.0	4.639891

(f)

t_i	$y_i(h = 0.5)$
1.0	2.5641
2.0	7.0835
3.0	17.6232
4.0	44.1189
5.0	115.5593

Section 9.2

4. $0.1(e - 1)$.

5. $h = 0.0116$.

Section 9.3

1. (a) $y_{i+1} = y_i + y_i^2[h^2(4t_i^2 y_i - 1) - 2ht_i]$.

   (b) $y_{i+1} = y_i + 3h(t_i - 1)(t_i + h - 1)$.

Section 9.4

1. (a)

t_i	$y_i(h = 0.1)$
0.2	2.6896
0.4	1.808487
0.6	1.216026
0.8	0.817656
1.0	0.549792

(c)

t_i	$y_i(h = 0.1)$
0.2	2.40485
0.4	1.836221
0.6	1.321514
0.8	0.885249
1.0	0.543803

2.

(a)

t_i	$y_i(h = 0.1)$
0.2	2.681297
0.4	1.797338
0.6	1.204800
0.8	0.807606
1.0	0.541358

(b)

t_i	$y_i(h = 0.1)$
0.2	1.639642
0.4	1.354902
0.6	1.137127
0.8	0.976082
1.0	0.862152
1.2	0.787500

3.

t_i	$y_i(h = 0.1)$	Error
1.2	1.764317	5.7E-3
1.4	2.551362	1.6E-2
1.6	4.079304	3.7E-2
1.8	7.177027	8.4E-2
2.0	13.850020	2.0E-1

4. (a) $y(1.25) \approx 2.07, \ E = 0.16.$

(b) $y(1.65) \approx 6.44, \ E = 1.8.$

5.

t_i	$y_i(h = 0.1)$
1.2	0.508918
1.4	0.532794
1.6	0.569412
1.8	0.618206
2.0	0.679570

$p(x) = 0.64 - 0.30t + 0.16t^2$.

(i) $y(1.02) \approx 4.99625,\ E = 4.7E - 4$.
(ii) $y(1.67) \approx 0.58470,\ E = 4.0E - 4$.
(iii) $y(1.98) \approx 0.67314,\ E = 3.0E - 4$.

6.

(a)

t_i	$y_i(h = 0.1)$
0.2	1.960397
0.4	1.846232
0.6	1.670540
0.8	1.452298
1.0	1.213061

(c)

t_i	$y_i(h = 0.2)$
1.4	1.4
1.8	1.8
3.0	3.0

(e)

t_i	$y_i(h = 0.5)$
1.0	3.17803
2.0	8.80537
3.0	22.07598
4.0	56.25298
5.0	149.13270

8. $y(0.5) \approx 1.648720,\ E = 6.3E - 7$.
$y(1.0) \approx 2.718281,\ E = 2.1E - 6$.

Section 9.6

1.

(a)

t_i	$y_i(h = 0.2)$
0.2	2.681600
0.4	1.797745
0.8	0.814563
1.2	0.372990
1.6	0.172457
2.0	0.082279

(c)

t_i	$y_i(h = 0.2)$
0.2	2.404192
0.6	1.318572
1.2	0.297438
1.6	0.059185
2.0	0.021230

(e)

t_i	$y_i(h = 0.1)$
0.2	1.960397
0.4	1.846252
0.6	1.670616
0.8	1.452436
1.0	1.213246

(f)

t_i	$y_i(h = 0.2)$
0.4	0.979225
0.8	0.850831
1.2	0.607661
1.6	0.353770
2.0	0.177025

2.

t_i	$y_i(h = 0.1)$	Error
0.4	0.268114	1.4E-5
0.6	0.329255	3.2E-5
0.8	0.359421	4.2E-5
1.0	0.367832	4.7E-5

Section 9.8

1.

(a)

t_i	$y_i(h = 0.1)$
0.4	1.797290
0.6	1.204704
0.8	0.807500
1.0	0.541258

(b)

t_i	$y_i(h = 0.1)$
0.4	1.354907
0.6	1.137137
0.8	0.976094
1.0	0.862164
1.2	0.787512

(c)

t_i	$y_i(h = 0.1)$
0.4	1.834524
0.6	1.318481
0.8	0.880768
1.0	0.538093

(d)

t_i	$y_i(h = 0.1)$
1.4	0.060102
1.6	0.011540
1.8	-0.003703
2.0	-0.088238

2.

t_i	$y_i(h = 0.1)$	Error
1.4	0.532793	1.2E-6
1.6	0.569410	2.4E-6
1.8	0.618203	3.0E-6
2.0	0.679567	3.5E-6

Section 9.10

1.

(a)

t_i	$y_i(h = 0.1)$
0.2	1.001403
0.4	1.011876
0.6	1.042759
0.8	1.109481
1.0	1.235013

(c)

t_i	$y_i(h = 0.1)$
0.2	-0.761336
0.4	-0.450754
0.6	-0.076667
0.8	0.351802
1.0	0.824356

2.

(a)

$t_i(h = 0.2)$	x_i	y_i
0.2	1.216733	0.270133
0.4	1.447219	0.728703
0.6	1.643529	1.466837
0.8	1.723895	2.605075
1.0	1.558194	4.297922

(c)

$t_i(h = 0.1)$	x_i	y_i
0.2	0.180796	0.873687
0.4	0.331208	0.872484
0.6	0.467719	0.969720
0.8	0.603843	1.146398
1.0	0.745363	1.388707

Section 9.11

1. (a)

$t_i(h = 0.1)$	x_i	y_i	$\lvert x_i - x(t_i)\rvert$	$\lvert y_i - y(t_i)\rvert$
1.0	-28.56	-3.11	-5.517	0.605

Section 9.12

1. $\frac{dx}{dt} = y, \quad \frac{dy}{dt} = -y + \sin x.$

2. $\frac{dx}{dt} = y, \quad \frac{dy}{dt} = (1 - x^2)y + x - x^3.$

3. $\frac{dx}{dt} = y, \quad \frac{dy}{dt} = \sin(x + y).$

4. $\frac{dx}{dt} = y, \quad \frac{dy}{dt} = -ty + x^2 \cos t.$

5. $y = \frac{dx}{dt} = 0, \quad x = 0,\ x = 1,\ x = -1$ so the fixed points are $(0, 0)$, $(0, 1)$, and $(0, -1)$.

7. $(0, 0)$.

8. $(0, 0)$, $(1, 1)$.

CHAPTER 10

Section 10.1

2.

(a)

x_i	$y_i(h = \pi/6)$
0	1.0
$\pi/6$	0.59805
$\pi/3$	-1.27953
$\pi/2$	-1.1E-16
$2\pi/3$	1.27953
$5\pi/6$	-0.59805
π	-1.0

(b)

x_i	$y_i(h = \pi/4)$
0	0.0
$\pi/4$	26.92366
$\pi/2$	37.72392
$3\pi/4$	26.22312
π	0.0

(c)

x_i	$y_i(h = 0.1)$
0.1	-0.05000
0.3	-0.11653
0.5	-0.13585
0.7	-0.10951
0.9	-0.04410
1.0	0.0

(d)

x_i	$y_i(h = 0.2)$
0.0	1.0
0.2	0.727356
0.4	0.498994
0.6	0.304895
0.8	0.139434
1.0	0.0

3.

x_i	$y_i(h = 0.1)$	Exact	Error
0.1	0.401034	0.4001	.0009
0.3	1.210314	1.2081	.0022
0.5	2.065237	2.0625	.0027
0.7	3.042561	3.0401	.0025
0.9	4.257267	4.2561	.0012
1.0	5.0	5.0	.0

Section 10.2

1.

(a)

x_i	$y_i(h = 0.2)$
0.2	0.778899
0.4	0.543991
0.6	0.321652
0.8	0.134837

(c)

x_i	$y_i(h = 0.2)$
1.1	-0.157272
1.3	-0.508495
1.5	-0.898971
1.7	-1.320660
1.9	-1.768007

2.

(a)

x_i	$y_i(h = 0.2)$
0.2	-0.74220
0.4	-0.53524
0.6	-0.35616
0.8	-0.18486

(c)

x_i	$y_i(h = 0.4)$
0.4	0.979910
0.8	1.113563
1.2	1.356478
1.6	1.661331

3.

x_i	$y_i(h = 0.1)$	Error
1.2	0.4375723075	5.7E-07
1.4	0.9421229809	7.2E-07
1.6	1.5040122213	6.1E-07
1.8	2.1160323388	3.5E-07

CHAPTER 11

Section 11.1

1. (a) $\lambda^2 - 4\lambda - 5$
   $\lambda_1 = -1$; $v^{(1)} = (-2, 1)^T$
   $\lambda_2 = 5$; $v^{(2)} = (4, 1)^T$

   (c) $\lambda^2 - 3\lambda + 2$
   $\lambda_1 = 1$; $v^{(1)} = (3/2, 1)^T$
   $\lambda_2 = 2$; $v^{(2)} = (1, 1)^T$

2. (a) $\lambda^2 - 5\lambda + 6 - a^2 - a$

6. (a) $\lambda^3 - 6\lambda^2 + 11\lambda - 6$
   $\lambda_1 = 2$; $v^{(1)} = (-2, 1, 4)^T$
   $\lambda_2 = 3$; $v^{(2)} = (-1, 1, 4)^T$
   $\lambda_3 = 1$; $v^{(3)} = (-1, 1, 2)^T$

(b) $\lambda^3 - 3\lambda^2 - 9\lambda + 27$
$\lambda_1 = 3;\ v^{(1)} = (1, 0, 1)^T$
$\lambda_2 = 3;\ v^{(2)} = (-1, 1, 0)^T$
$\lambda_3 = -3;\ v^{(3)} = (-1, -1, 1)^T$

Section 11.2

1. (a) $\lambda = 7.00251001,\ \mathbf{x} = [1, 0.666726, 5.97 \times 10^{-5}]^T$

   (b) $\lambda = 5,\ \mathbf{x} = [0.121826, 1, 0.121826]^T$

   (c) $\lambda = 7.23117887,\ \mathbf{x} = [0.173101, 1, 0.035389, -0.058309]^T$

   (d) $\lambda = 8.16062223,\ \mathbf{x} = [-0.234068, 0.524851, -0.106132, 1]^T$

Section 11.3

1. (b) $\lambda_1 = -1;\ v^{(1)} = (-2, 1, 1)^T$
   $\lambda_2 = 5;\ v^{(2)} = (1, 1, 1)^T$
   $\lambda_3 = -3;\ v^{(3)} = (0, -1, 1)^T$

   (c) $\lambda_1 = 5;\ v^{(1)} = (0, 1, 0)^T$
   $\lambda_2 = 5;\ v^{(2)} = (1, 0, 2)^T$
   $\lambda_3 = -10;\ v^{(3)} = (-2, 0, 1)^T$

   (d) $\lambda_1 = 3;\ v^{(1)} = (-1, 0, 1)^T$
   $\lambda_2 = -3;\ v^{(2)} = (-1, 0, 1)^T$
   $\lambda_3 = 0;\ v^{(3)} = (1, -1, 1)^T$

2. $\lambda = 3$

3. $X(t) = Ae^{5.10t}[1, 1.019, 1.032]^T + Be^{-2.85t}[-0.013, 1, -0.975]^T +$
   $Ce^{-0.95t}[-1.998, 0.949, 1]^T$

4. $\lambda_1 = 3;\ v^{(1)} = (1, 1, 0, -1, -1)^T$
   $\lambda_2 = 1.5359;\ v^{(2)} = (.5774, 1, 1.1547, 1, .5774)^T$
   $\lambda_3 = 5;\ v^{(3)} = (-1, 0, 1, 0, -1)^T$
   $\lambda_4 = 8.4641;\ v^{(4)} = (-.5774, 1, -1.1547, 1, -.5774)^T$
   $\lambda_5 = 7;\ v^{(5)} = (-1, 1, 0, -1, 1)^T$

Section 11.4

1. $k_1 = \pm 1.5883, k_2 = \pm 2.8284, k_3 = \pm 3.6711$

Section 11.5

1. (a) $x(x^2 + 2x + r) = 0$ so, $x = 0$ and $x = -1 \pm \sqrt{(1+r)}$

   (d) $x(r + x^2) = 0$ so $x = 0$ is the only fixed point. There is no bifurcation.

2. For small values of ω, $(0, 0)$ is stable while for larger it is unstable.

CHAPTER 12

Section 12.1

3. (a)

$t\backslash x$	0.0	0.2	0.4	0.6	0.8	1.0
0.10	0.00	0.141066	0.219796	0.209357	0.124175	0.00

(b)

$t\backslash x$	0.0	0.2	0.4	0.6	0.8	1.0
0.20	0.00	0.196328	0.262656	0.317734	0.162344	0.00

(c)

$t\backslash x$	0.0	20	40	60	80	100
0.05	0.00	2.35492	49.95245	49.95245	2.35492	0.00

4. (b)

$t\backslash x$	0.0	0.2	0.4	0.6	0.8	1.0
0.01	0.00	0.532286	0.861257	0.861257	0.532286	0.00
Error	0.00	0.0004	0.0007	0.0007	0.0004	0.00

5. $\frac{u_{i+1}^{k+1} - 2u_{i+1}^{k} + u_{i+1}^{k-1}}{(\Delta x)^2} = u_i^k \left[\frac{u_{i+1}^k - u_i^k}{\Delta y} \right]$

Section 12.2

2. (a)

$t\backslash x$	0.0	0.2	0.4	0.6	0.8	1.0
0.05	0.00	0.202000	0.408000	0.618000	0.832000	0.00
0.50	0.00	0.252190	0.473242	-0.056159	-0.119455	0.00

(b)

$t\backslash x$	0.0	0.63	1.26	1.88	2.51	3.14
0.10	0.00	0.998943	1.501598	1.501598	0.998943	0.00
1.00	0.00	0.482248	0.880878	0.880878	0.482248	0.00

(c)

$t\backslash x$	0.0	0.25	0.5	0.75	1.0
0.25	0.00	0.176777	0.250000	0.176777	0.00
1.00	0.000	1.000000	-0.000000	-1.000000	0.00

Section 12.3

1.

(a)

$y\backslash x$	0.0	0.50	1.00	1.50	2.00
0.50	0.0000	0.7142	1.8749	4.2857	10.0000
1.00	0.0000	0.9820	2.4999	5.2678	10.0000
1.50	0.0000	0.7142	1.8750	4.2857	10.0000
2.00	0.0000	0.0000	0.0000	0.0000	10.0000

(b)

$y\backslash x$	0.0	0.79	1.57	2.36	3.14
0.79	0.0000	0.0583	0.1508	0.3318	0.7071
1.57	0.0000	0.0825	0.2133	0.4692	1.0000
2.36	0.0000	0.0583	0.1509	0.3318	0.7071
3.14	0.0000	0.0000	0.0000	0.0000	0.0000

(c)

$y\backslash x$	0.0	0.25	0.50	0.75	1.00
0.79	0.5000	0.2633	0.1249	0.0491	0.0000
1.57	1.0000	0.4285	0.1874	0.0714	0.0000
2.36	0.5000	0.2634	0.1250	0.0491	0.0000
3.14	0.0000	0.0000	0.0000	0.0000	0.0000

(d)

$y\backslash x$	0.0	0.25	0.50	0.75	1.00
0.25	0.0000	0.0150	0.0390	0.0865	0.1875
0.50	0.0000	0.0211	0.0546	0.1194	0.2500
0.75	0.0000	0.0150	0.0390	0.0865	0.1875
1.00	0.0000	0.0000	0.0000	0.0000	0.0000

Index

A

Absolute error, 27
Accuracy
    decimal digits, 28
    significant digit, 28
Adams methods
    Adams-Bashforth method, 273
    Adams-Moulton method, 283
Algorithm list, 401
Arithmetic
    interval, 30
    floating point, 21
Augmented matrix, 86

B

Backward difference, 136
Back substitution, 84
Biased exponent, 24
Basis function
    cubic-spline, 161
    piecewise linear, 153
Bifurcation, 352
Binary numbers, 22
Bisection method, 37
Bit, 22
Boundary-value problems, 311
Break points, 155

C

Cardinal functions, 147
Central divided differences, 205
Characteristic polynomial, 329
Cholesky's method, 103
Chopping arithmetic, 26
Coefficient matrix, 115
Composite Simpson's rule, 222
Composite Trapezoid rule, 215
Convergence
    Jacobi method, 123
    Newton's method, 58
    Rate, 58
    Secant method, 58
    Seidel method, 123
Continuation methods, 340
Corrector methods, 282
Crank-Nicolson method, 371
Crout's method, 103
Cubic spline, 161
Curve fitting, 175

D

Decimal machine number, 22
Deflation technique, 338
Derivatives, 204
Determinant of matrix, 329
Diagonal matrix, 80

Difference equation, 350
Differential equations, IVP *see* Initial-value problems
Differential equations, BVP *see* Boundary-value problems
Differentiation *see* Numerical differentiation
Diffusion equation, 359
Digit significant, 28
Direct factorization, 79
Direct methods, 115
Dirichlet boundary conditions, 380
Divided difference, 135
Doolittle's method, 103
Double precision, 28

E

Finite differences
  derivative formulas, 204
  PDEs, 350
  BVP, 311
Fixed-point iteration, 60
Floating point representation
  double precision, 28
  IEEE normalized, 24
  overflow, 23
Forward difference, 138
Forward substitution, 102
Fourth-order Adams-Bashforth method, 275
Fourth-order Adams-Moulton method, 283
Fraction binary, 22
Fundamental Theorem of Calculus, 394