Introduction to Numerical Computation in Pascal

Macmillan Computer Science Series

Consulting Editor

Professor F. H. Sumner, University of Manchester

- S.T. Allworth and R.N. Zobel, *Introduction to Real-Time Software Design*, second edition
Ian O. Angell, *A Practical Introduction to Computer Graphics*
Ian O. Angell and Gareth Griffith, *High Resolution Graphics Using Fortran 77*
R.E. Berry and B.A.E. Meekings, *A Book on C*
G.M. Birtwistle, *Discrete Event Modelling on Simula*
T.B. Boffey, *Graph Theory in Operations Research*
Richard Bornat, *Understanding and Writing Compilers*
J.K. Buckle, *Software Configuration Management*
W.D. Burnham and A.R. Hall, *Prolog Programming and Applications*
J.C. Cluley, *Interfacing to Microprocessors*
Robert Cole, *Computer Communications*, second edition
Derek Coleman, *A Structured Programming Approach to Data*
Andrew J.T. Colin, *Fundamentals of Computer Science*
Andrew J.T. Colin, *Programming and Problem-solving in Algol 68*
S.M. Deen, *Fundamentals of Data Base Systems*
S.M. Deen, *Principles and Practice of Data Base Systems*
B.T. Denvir, *Introduction to Discrete Mathematics for Software Engineering*
P.M. Dew and K.R. James, *Introduction to Numerical Computation in Pascal*
M.R.M. Dunsmuir and G.J. Davies, *Programming the UNIX System*
K.C.E. Gee, *Introduction to Local Area Computer Networks*
J.B. Gosling, *Design of Arithmetic Units for Digital Computers*
Roger Hutton, *Fortran for Students*
Roger Hutton, *Z80 Assembly Language Programming for Students*
Roland N. Ibbett, *The Architecture of High Performance Computers*
Patrick Jaulent, *The 68000 – Hardware and Software*
J.M. King and J.P. Pardoe, *Program Design Using JSP – A Practical Introduction*
H. Kopetz, *Software Reliability*
E.V. Krishnamurthy, *Introductory Theory of Computer Science*
V.P. Lane, *Security of Computer Based Information Systems*
Graham Lee, *From Hardware to Software: an introduction to computers*
A.M. Lister, *Fundamentals of Operating Systems*, third edition
G.P. McKeown and V.J. Rayward-Smith, *Mathematics for Computing*
Brian Meek, *Fortran, PL/I and the Algols*
Barry Morrell and Peter Whittle, *CP/M 80 Programmer's Guide*
Derrick Morris, *System Programming Based on the PDP11*
Pim Oets, *MS-DOS and PC-DOS – A Practical Guide*
Christian Queinnec, *LISP*
W.P. Salman, O. Tisserand and B. Toulout, *FORTH*
L.E. Scales, *Introduction to Non-linear Optimization*
Peter S. Sell, *Expert Systems – A Practical Introduction*
Colin J. Theaker and Graham R. Brookes, *A Practical Course on Operating Systems*
J-M. Trio, *8086-8088 Architecture and Programming*
M.J. Usher, *Information Theory for Information Technologists*
Colin Walls, *Programming Dedicated Microprocessors*
B.S. Walker, *Understanding Microprocessors*
Peter J.L. Wallis, *Portable Programming*
I.R. Wilson and A.M. Addyman, *A Practical Introduction to Pascal – with BS6192*, second edition

Introduction to Numerical Computation in Pascal

P. M. Dew and K. R. James

*Department of Computer Studies,
University of Leeds*

M
MACMILLAN

© P. M. Dew and K. R. James 1983

All rights reserved. No reproduction, copy or transmission of this publication may be made without written permission.

No paragraph of this publication may be reproduced, copied or transmitted save with written permission or in accordance with the provisions of the Copyright Act 1956 (as amended).

Any person who does any unauthorised act in relation to this publication may be liable to criminal prosecution and civil claims for damages.

First published 1983

Reprinted 1983 (with minor corrections), 1985, 1986

Published by
MACMILLAN EDUCATION LTD
Houndmills, Basingstoke, Hampshire RG21 2XS
and London
Companies and representatives
throughout the world

ISBN 978-0-333-32897-2

ISBN 978-1-349-17097-5 (eBook)

DOI 10.1007/978-1-349-17097-5

Contents

<i>Preface</i>	viii
Introduction	x
PART 1 FUNDAMENTAL TECHNIQUES	1
1 Programming in Pascal	3
1.1 Basic Data Types	3
1.2 Input/output	8
1.3 Subprograms	9
1.4 Library Routines	13
1.5 Further Data Types	17
*1.6 Graphical Output: procedure graf	21
2 Principles of Mathematical Software	28
2.1 Design and Analysis of Algorithms	28
2.2 Software Realisation: The mathlib Library	31
2.3 Numerical Testing and Documentation	36
*2.4 Variable-size (Conformant) Arrays	40
3 Basic Mathematics and Applications	44
3.1 Sets and Functions	44
3.2 Continuous Functions	48
3.3 Derivatives and Polynomials	52
3.4 Linear Approximations	55
3.5 Interpolation	57
3.6 Sequences and Series	60
3.7 Taylor's Series	63
4 Real Numbers, Machine Numbers and Errors	70
4.1 Errors and their Sources	70
4.2 Measurement and Control of Error	72
4.3 Machine Numbers and mathlib Constants	76
4.4 Arithmetic on a 4-digit Computer	81
*4.5 Error Propagation	85
4.6 Numerical Instability of Algorithms	88

PART 2 DEVELOPMENT OF MATHEMATICAL SOFTWARE	93
5 Solution of Non-linear Equations in One Variable:	
Newton's Method	95
5.1 Roots of Equations	95
5.2 Newton's Method	98
5.3 Implementation of Newton's Method	101
5.4 Newton's Method for Polynomials: procedure polnewt	106
5.5 Order of Convergence	114
6 Solution of Non-linear Equations in One Variable:	
Interval Methods	120
6.1 Interval Code: procedure bisect	120
6.2 Alternatives to Newton's Method	125
6.3 Improved Interval Code: procedure bisecant	128
6.4 Battery Testing of Interval Codes	133
*6.5 Rational Interpolation: procedure bisrat	134
7 Systems of Linear Equations	140
7.1 Basic Concepts	140
7.2 Triangular Factorisation	147
7.3 Solving Linear Systems	155
7.4 Condition and Norms	164
7.5 Large Linear Systems	171
*7.6 The Conjugate Gradient Method	178
8 Numerical Integration: Fixed-point Rules	189
8.1 The Trapezoidal Rule	190
8.2 Simpson's Rule	197
8.3 Gaussian Rules	202
8.4 Comparison of Integration Rules	208
9 Numerical Integration: Adaptive Methods	213
9.1 Outline of an Automatic Integrator based on the Trapezoidal Rule	213
9.2 Romberg Integration	218
*9.3 Implementation of Romberg Integration	221
9.4 Adaptive Simpson's Rule	236
*9.5 Transformation Methods	247
<i>Appendix A: Predeclared Mathematical Functions in Pascal</i>	261
<i>Appendix B: Summary of the mathlib Library</i>	262
<i>Appendix C: The Utility Functions powerr, poweri and roundt</i>	272
<i>Appendix D: Notes on Portability</i>	275

<i>Contents</i>	<i>vii</i>
<i>Solutions to Exercises</i>	279
<i>References</i>	285
<i>Index</i>	287

*Sections marked with an asterisk may be read superficially without loss of continuity on first reading.

Preface

Our intention in this book is to cover the core material in numerical analysis normally taught to students on degree courses in computer science. The main emphasis is placed on the use of analysis and programming techniques to produce well-designed, reliable mathematical software. The treatment should be of interest also to students of mathematics, science and engineering who wish to learn how to write good programs for mathematical computations.

The reader is assumed to have some acquaintance with Pascal programming. Aspects of Pascal particularly relevant to numerical computation are revised and developed in the first chapter. Although Pascal has some drawbacks for serious numerical work (for example, only one precision for real numbers), the language has major compensating advantages: it is a widely used teaching language that will be familiar to many students and it encourages the writing of clear, well-structured programs. By careful use of structure and documentation, we have produced codes that we believe to be readable; particular care has been taken to ensure that students should be able to understand the codes in conjunction with the descriptive material given in the book.

Our aim, unusual in an introductory text, is to explain in some depth good-quality mathematical software, paying full attention (subject to the overriding aim of clarity) to robustness, adaptability and error control as well as to the theory underlying the methods. The central theme is the description of a Pascal mathematical library, *mathlib*, which we have designed primarily for teaching purposes and used for computing courses at Leeds University. This gives us a well-defined framework for the development of theory and algorithms, exposes some important problems of writing mathematical software, and provides a consistent thread of motivation and reinforcement. Many of the exercises and projects make use of the library or require the student to modify or extend the library codes.

The book can also be used as a source of up-to-date mathematical routines in Pascal which are not readily available elsewhere. It is hoped that this will make it useful to practitioners and hobbyists with Pascal on their own micros. All the routines have been tested on the Amdahl 470 V/7 and DECsystem-10 computers at Leeds University, and library subsets have been mounted successfully on an Apple microprocessor. In the interests of portability, we have adhered in our coding as far as possible to the Pascal standard defined by Jensen and Wirth (portability of the library is discussed in appendix D). As a teaching aid,

versions of the library are available for mainframe computers and for some microprocessors (further information is given below).

A particular debt is due to each of the following, who read all or part of the manuscript, in some cases more than once, and gave us many valuable comments: Dr W. Riha, Dr I. Gladwell, Dr D. Sayers, Dr J. K. Reid, Dr Mary Almond, Dr Ruth Thomas, Mr K. Hopper. The computer printouts used in the book were produced with the kind co-operation of Dr A. A. Hock and Miss C. Redford of the University of Leeds Computing Service. It is a pleasure also to record our thanks to the diligent typists in the Department of Computer Studies at Leeds University, Audrey Abbott and her helpers.

Leeds, 1982

P.M.D.

K.R.J.

NOTE ON AVAILABILITY OF THE MATHLIB LIBRARY

The mathlib library as described in this book is available for distribution on mainframe computers. A version of the library has also been written to run under UCSD Pascal on the Apple II microcomputer. It is planned to make the library available on other microcomputers in the future. Further details and distribution costs can be obtained by writing to the Editor of the Computer Science Series at the Macmillan Press.

Introduction

The aim of this book is to develop the mathematical and computer programming skills required to understand and write reliable, efficient programs for some of the more important mathematical computations. We refer to a collection of programs of this type as *mathematical software*.

The need for mathematical software is evident if we consider that, to investigate phenomena in the physical or social sciences, it is necessary to represent systems by mathematical models. For example, dynamical systems and electrical networks are often modelled in terms of differential equations. An essential feature of any such model is that it is only an approximation to the real-life situation; the investigator tries to discover the most significant terms and include these in his model. Even so, it is rarely possible to obtain an exact solution. Instead, the problem is normally reduced to a number of basic mathematical tasks, for example, the solution of simultaneous equations or the evaluation of definite integrals, which can be solved *numerically* to a specified accuracy. It is desirable to provide the modeller with a set of standard programs to perform these basic computations reliably and efficiently.

From the earliest days of computing it has been recognised that there is a need for *library routines* to perform the basic mathematical computations. Originally, a library was a collection of programs written by users and thought to be of general interest. The programs were expected to run correctly only in the particular computing environment of the installation in which they had been developed. As there are many different computer systems in operation and the lifetime of any one system is comparatively short, this was found to be very wasteful of time and effort.

Today there are a number of projects that aim to provide comprehensive collections of mathematical software in a form which is, as far as possible, independent of the computing environment (machine-independent). The objective is to develop routines which can be used on a variety of computers, compilers and operating systems, and to supply users with well-defined program specifications so that they can employ the programs sensibly and correctly. For example, in Britain in the early 1970s the Numerical Algorithms Group (NAG) was established, initially to provide British universities with a set of well-tried mathematical routines on the ICL 1900 series of computers. Subsequently the NAG library has been made available on a number of machine ranges and is sold as a commercial product.

A library of routines, regularly updated and maintained, has the great advantage that the latest developments in theory and practice can be passed on without requiring from the user any specialised knowledge of the subject. Indeed, a modern computer library of mathematical routines represents the 'state of the art' in numerical computation.

Much of the mathematical software used in practice is written in FORTRAN 66. One reason is that programs written in this language are portable; that is, they can be used on many computer systems with little modification. FORTRAN, however, does not lend itself easily to structured programming and is probably not a good vehicle for teaching programming skills[†]. Since our aim is to present programs that are well-structured and (we hope) easy to read, we use the modern programming language Pascal.

We shall consider a number of mathematical problems that are of basic importance, develop the underlying theory and, in the light of this, discuss how to write suitable library routines to perform the necessary computations. The routines are collected together in a Pascal library called *mathlib*, a summary of which is given in appendix B. The intention is that the reader should be able to understand as well as use the routines and, for this reason, some of the finer detail which would be included in a commercial library has been simplified in the interests of clarity.

[†]FORTRAN in its essentials is one of the oldest programming languages. More recent versions, for example FORTRAN 77, contain some facilities for structured programming.