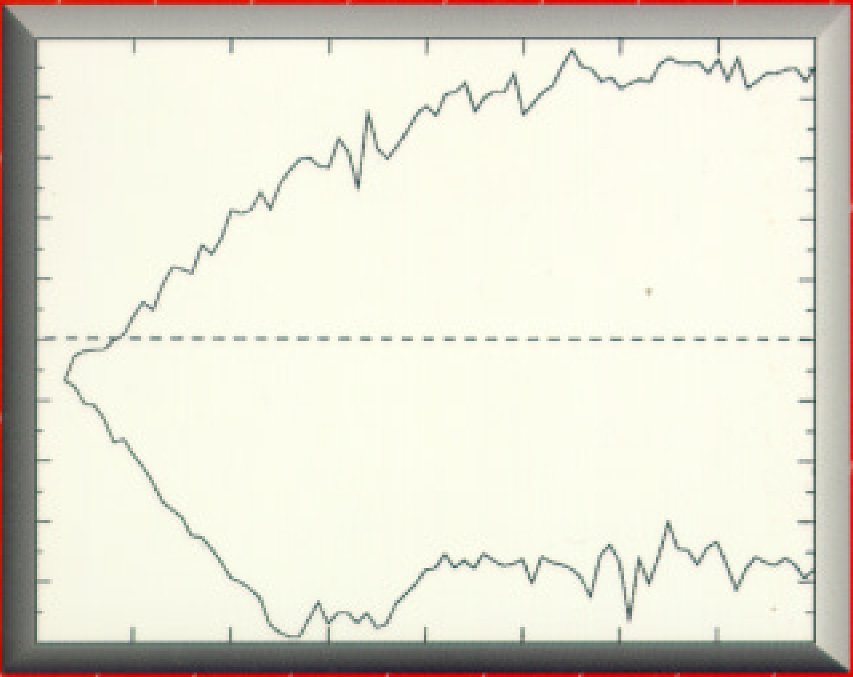
Home Next

Nicholas J. Higham

Accuracy and Stability of Numerical Algorithms



siam®

Accuracy and Stability of Numerical Algorithms

Nicholas J. Higham

University of Manchester
Manchester, England

Accuracy and Stability of Numerical Algorithms

Society for Industrial and Applied Mathematics
Philadelphia

Copyright © 1996 by the Society for Industrial and Applied Mathematics.

1 0 9 8 7 6 5 4 3 2

All rights reserved. Printed in the United States of America. No part of this book may be reproduced, stored, or transmitted in any manner without the written permission of the publisher. For information. write to the Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688.

Library of Congress Cataloging-in-Publication Data

Higham, Nicholas J., 1961-
Accuracy and stability of numerical algorithms / Nicholas J. Higham.
p. cm.
Includes bibliographical references (p. -) and index.
ISBN 0-89871-355-2 (pbk.)
1. Numerical analysis--Data processing. 2. Computer algorithms.
I. Title.
QA297.H53 1996
519.4'0285'51 --dc20 **95-39903**

siam is a registered trademark.

Dedicated to

Alan M. Turing
and
James H. Wilkinson

Contents

List of Figures xvii

List of Tables xix

Preface xxi

About the Dedication xxvii

1 Principles of Finite Precision Computation 1
- 1.1 Notation and Background 2
- 1.2 Relative Error and Significant Digits 4
- 1.3 Sources of Errors 5
- 1.4 Precision Versus Accuracy 7
- 1.5 Backward and Forward Errors 7
- 1.6 Conditioning 9
- 1.7 Cancellation 10
- 1.8 Solving a Quadratic Equation 11
- 1.9 Computing the Sample Variance 12
- 1.10 Solving Linear Equations 13
  - 1.10.1 GEPP Versus Cramer's Rule 14
- 1.11 Accumulation of Rounding Errors 16
- 1.12 Instability Without Cancellation 17
  - 1.12.1 The Need for Pivoting 17
  - 1.12.2 An Innocuous Calculation? 17
  - 1.12.3 An Infinite Sum 18
- 1.13 Increasing the Precision 19
- 1.14 Cancellation of Rounding Errors 21
  - 1.14.1 Computing $(e^x - 1)/x$ 22
  - 1.14.2 QR Factorization 24
- 1.15 Rounding Errors Can Be Beneficial 26
- 1.16 Stability of an Algorithm Depends on the Problem 27
- 1.17 Rounding Errors Are Not Random 29
- 1.18 Designing Stable Algorithms 30
- 1.19 Misconceptions 31

1.20 Rounding Errors in Numerical Analysis 32
1.21 Notes and References 32
Problems 36

2 Floating Point Arithmetic **39**
2.1 Floating Point Number System 40
2.2 Model of Arithmetic 44
2.3 IEEE Arithmetic 45
2.4 Aberrant Arithmetics 48
2.5 Choice of Base and Distribution of Numbers 51
2.6 Statistical Distribution of Rounding Errors 52
2.7 Alternative Number Systems 53
2.8 Accuracy Tests 54
2.9 Notes and References 56
Problems 62

3 Basics **67**
3.1 Inner and Outer Products 68
3.2 The Purpose of Rounding Error Analysis 71
3.3 Running Error Analysis 72
3.4 Notation for Error Analysis 73
3.5 Matrix Multiplication 76
3.6 Complex Arithmetic 78
3.7 Miscellany 80
3.8 Error Analysis Demystified 82
3.9 Other Approaches 83
3.10 Notes and References 84
Problems 84

4 Summation **87**
4.1 Summation Methods 88
4.2 Error Analysis 89
4.3 Compensated Summation 92
4.4 Other Summation Methods 97
4.5 Statistical Estimates of Accuracy 98
4.6 Choice of Method 98
4.7 Notes and References 100
Problems 100

5 Polynomials **103**
5.1 Horner‘s Method 104
5.2 Evaluating Derivatives 106
5.3 The Newton Form and Polynomial Interpolation . . . , ... 109

5.4 Notes and References 113
Problems 114

6 Norms 117
6.1 Vector Norms 118
6.2 Matrix Norms 120
6.3 The Matrix *p*-Norm 124
6.4 Notes and References 126
Problems 127

7 Perturbation Theory for Linear Systems 131
7.1 Normwise Analysis 132
7.2 Componentwise Analysis 134
7.3 Scaling to Minimize the Condition Number 137
7.4 The Matrix Inverse 140
7.5 Extensions 140
7.6 Numerical Stability 141
7.7 Practical Error Bounds 142
7.8 Perturbation Theory by Calculus 144
7.9 Notes and References 145
Problems 147

8 Triangular Systems 151
8.1 Backward Error Analysis 152
8.2 Forward Error Analysis 155
8.3 Bounds for the Inverse 159
8.4 A Parallel Fan-In Algorithm 162
8.5 Notes and References 164
8.5.1 LAPACK 166
Problems 166

9 LU Factorization and Linear Equations 169
9.1 Gaussian Elimination 170
9.2 Error Analysis 174
9.3 The Growth Factor 177
9.4 Special Matrices 181
9.5 Tridiagonal Matrices 183
9.6 Historical Perspective 186
9.7 Scaling 191
9.8 A Posteriori Stability Tests 192
9.9 Sensitivity of the LU Factorization 194
9.10 Notes and References 195
9.10.1 LAPACK 198

Problems 199

10 Cholesky Factorization 203
10.1 Symmetric Positive Definite Matrices 204
10.1.1 Error Analysis 205
10.2 Sensitivity of the Cholesky Factorization 209
10.3 Positive Semidefinite Matrices 210
10.3.1 Perturbation Theory 211
10.3.2 Error Analysis 214
10.4 Symmetric Indefinite Matrices and Diagonal Pivoting Method. 218
10.4.1 Complete Pivoting 219
10.4.2 Partial Pivoting 221
10.5 Nonsymmetric Positive Definite Matrices 223
10.6 Notes and References 224
10.6.1 LAPACK 226
Problems 227

11 Iterative Refinement 231
11.1 Convergence of Iterative Refinement 232
11.2 Iterative Refinement Implies Stability 235
11.3 Notes and References 241
11.3.1 LAPACK. 243
Problems 243

12 Block LU Factorization 245
12.1 Block Versus Partitioned LU Factorization 246
12.2 Error Analysis of Partitioned LU Factorization 248
12.3 Error Analysis of Block LU Factorization 250
12.3.1 Block Diagonal Dominance 251
12.3.2 Symmetric Positive Definite Matrices 255
12.4 Notes and References 257
12.3.1 LAPACK 258
Problems 258

13 Matrix Inversion 261
13.1 Use and Abuse of the Matrix Inverse 262
13.2 Inverting a Triangular Matrix 265
13.2.1 Unblocked Methods 265
13.2.2 Block Methods 267
13.3 Inverting a Full Matrix by LU Factorization 270
13.3.1 Method A 270
13.3.2 Method B 271

        13.3.3 Method C 272
        13.3.4 Method D 273
        13.3.5 Summary. 275
    13.4 Gauss-Jordan Elimination 275
    13.5 The Determinant 281
        13.5.1 Hyman's Method 282
    13.6 Notes and References 283
        13.6.1 LAPACK 285
        Problems 285

14 Condition Number Estimation **289**
    14.1 How to Estimate Componentwise Condition Numbers 290
    14.2 The p-Norm Power Method. 291
    14.3 LAPACK 1-Norm Estimator 294
    14.4 Other Condition Estimators 297
    14.5 Condition Numbers of Tridiagonal Matrices 301
    14.6 Notes and References 304
        14.6.1 LAPACK 306
        Problems 306

15 The Sylvester Equation **309**
    15.1 Solving the Sylvester Equation 311
    15.2 Backward Error 313
        15.2.1 The Lyapunov Equation 316
    15.3 Perturbation Result 318
    15.4 Practical Error Bounds 320
    15.5 Extensions 321
    15.6 Notes and References 322
        15.6.1 LAPACK 324
        Problems 324

16 Stationary Iterative Methods **325**
    16.1 Survey of Error Analysis 327
    16.2 Forward Error Analysis 329
        16.2.1 Jacobi's Method 332
        16.2.2 Successive Overrelaxation 334
    16.3 Backward Error Analysis 334
    16.4 Singular Systems 336
        16.4.1 Theoretical Background 336
        16.4.2 Forward Error Analysis 338
    16.5 Stopping an Iterative Method 341
    16.6 Notes and References 343

Problems 343

17 Matrix Powers **345**
17.1 Matrix Powers in Exact Arithmetic 346
17.2 Bounds for Finite Precision Arithmetic 353
17.3 Application to Stationary Iteration 358
17.4 Notes and References 358
Problems 359

18 *QR* Factorization **361**
18.1 Householder Transformations 362
18.2 *QR* Factorization 363
18.3 Error Analysis of Householder Computations 364
18.4 Aggregated Householder Transformations 370
18.5 Givens Rotations 371
18.6 Iterative Refinement 375
18.7 Gram-Schmidt Orthogonalization 376
18.8 Sensitivity of the *QR* Factorization 381
18.9 Notes and References 383
18.9.1 LAPACK 386
Problems 387

19 The Least Squares Problem **391**
19.1 Perturbation Theory 392
19.2 Solution by *QR* Factorization 395
19.3 Solution by the Modified Gram-Schmidt Method 396
19.4 The Normal Equations 397
19.5 Iterative Refinement 399
19.6 The Seminormal Equations 403
19.7 Backward Error 404
19.8 Proof of Wedin's Theorem 407
19.9 Notes and References 409
19.9.1 LAPACK 412
Problems 412

20 Underdetermined Systems **415**
20.1 Solution Methods 416
20.2 Perturbation Theory 417
20.3 Error Analysis 419
20.4 Notes and References 422
20.4.1 LAPACK 423
Problems 423

21 Vandermonde Systems 425

21.1 Matrix Inversion 426
21.2 Primal and Dual Systems 428
21.3 Stability 434
    21.3.1 Forward Error 435
    21.3.2 Residual 437
    21.3.3 Dealing with Instability 438
21.4 Notes and References 440
    Problems 441

22 Fast Matrix Multiplication 445

22.1 Methods 446
22.2 Error Analysis 450
    22.2.1 Winograd's Method 451
    22.2.2 Strassen's Method 452
    22.2.3 Bilinear Noncommutative Algorithms 456
    22.2.4 The 3M Method 458
22.3 Notes and References 459
    Problems 461

23 The Fast Fourier Transform and Applications 465

23.1 The Fast Fourier Transform 466
23.2 Circulant Linear Systems 468
23.3 Notes and References 470
    Problems 471

24 Automatic Error Analysis 473

24.1 Exploiting Direct Search Optimization 474
24.2 Direct Search Methods 477
24.3 Examples of Direct Search 479
    24.3.1 Condition Estimation 480
    24.3.2 Fast Matrix Inversion 481
    24.3.3 Solving a Cubic 483
24.4 Interval Analysis 485
24.5 Other Work 487
24.6 Notes and References 489
    Problems 490

25 Software Issues in Floating Point Arithmetic 491

25.1 Exploiting IEEE Arithmetic 492
25.2 Subtleties of Floating Point Arithmetic 495
25.3 Cray Peculiarities 496
25.4 Compilers 497

25.5 Determining Properties of Floating Point Arithmetic 497
25.6 Testing a Floating Point Arithmetic 498
25.7 Portability 499
  25.7.1 Arithmetic Parameters 499
  25.7.2 2×2 Problems in LAPACK 500
  25.7.3 Numerical Constants 501
  25.7.4 Models of Floating Point Arithmetic 501
25.8 Avoiding Underflow and Overflow 502
25.9 Multiple Precision Arithmetic 504
25.10 Patriot Missile Software Problem 506
25.11 Notes and References 507
  Problems 508

26 A Gallery of Test Matrices 513
26.1 The Hilbert and Cauchy Matrices 514
26.2 Random Matrices 517
26.3 “Randsvd” Matrices 519
26.4 The Pascal Matrix 520
26.5 Tridiagonal Toeplitz Matrices 524
26.6 Companion Matrices 525
26.7 Notes and References 526
  26.7.1 LAPACK 527
  Problems 527

A Solutions to Problems 529

B Singular Value Decomposition, *M*-Matrices 579
B.1 Singular Value Decomposition 580
B.2 *M*-Matrices 580

C Acquiring Software 581
C.1 Internet. 582
C.2 Netlib 582
C.3 MATLAB 583
C.4 NAG Library and FTN90 Compiler 583

D Program Libraries 585
D.1 Basic Linear Algebra Subprograms 586
D.2 EISPACK 587
D.3 LINPACK 587
D.3 LAPACK 587
  D.4.1 Structure of LAPACK 588

E The Test Matrix Toolbox 591

Bibliography 595

Name Index 665

Subject Index 675

List of Figures

1.1 Backward and forward errors for $y = f(x)$. 8

1.2 Mixed forward-backward error for $y = f(x)$. 9

1.3 Forward errors $\|x - \widehat{x}\|_\infty/\|x\|_\infty$ and relative residuals $\|b - A\widehat{x}\|_\infty/(\|A\|_\infty\|\widehat{x}\|_\infty)$ versus precision. 20

1.4 Absolute error versus precision, t = $-\log_2 u$ 21

1.5 Relative errors $\|A_k - \hat{A}_k\|_2/\|A\|_2$ for Givens QR factorization. 25

1.6 Values of rational function $r(x)$ computed by Horner's rule. 29

2.1 Relative distance from x to the next larger machine number (β=2, t=24), displaying wobbling precision. 44

4.1 Recovering the rounding error. 92

4.2 Errors $|y(1) - \widehat{y}_n|$ for Euler's method with and without compensated summation. 96

5.1 Computed polynomial values and running and a priori bounds for Horner's method. 107

6.1 Plots of p versus $||A||p$, for $1 \le p \le 15$. 125

9.1 A banded matrix. 182

13.1 Residuals for inverses computed by MATLAB'S INV function. 264

14.1 Underestimation ratio for Algorithm 14.4 for 5×5 matrix $A(\theta)$. 297

16.1 Forward and backward errors for SOR iteration. 327

17.1 A typical hump for a convergent, nonnormal matrix. 347

17.2 Diverging powers of a nilpotent matrix, C_{14} 347

17.3 Infinity norms of powers of 2×2 matrix J in (17.2). 349

17.4 Computed powers of chebspec matrices. 356

17.5 Pseudospectra for chebspec matrices. 357

17.6 Pseudospectrum for SOR iteration matrix. 359

18.1 Householder matrix P times vector x. 363
18.2 Givens rotation, $y = G(i,j,\theta)x$. 372

22.1 Exponent versus time for matrix multiplication. 449
22.2 Errors for Strassen's method with two random matrices of dimension $n = 1024$. 457

23.1 Error in FFT followed by inverse FFT. 468

24.1 The possible steps in one iteration of the MDS method when n=2. 478

25.1 Rational function r 493
25.2 Error in evaluating rational function r 494

26.1 spy(rem(pascal(32),2)). 524
26.2 Pseudospectra of compan(A). , . . . 526
26.3 Pseudospectra of 32 × 32 pentadiagonal Toeplitz matrices. 528

List of Tables

1.1 Computed approximations $\widehat{f}_n = fl((1+1/n)^n)$ to $e = 2.71828$ 16
1.2 Computed values of $(e^x - 1)/x$ from Algorithms 1 and 2. . . . 23
1.3 Results from GE without pivoting on an upper Hessenberg matrix 28

2.1 Floating point arithmetic parameters. 41
2.2 IEEE arithmetic exceptions and default results. 46
2.3 Test arithmetics. 54
2.4 Sine test. 55
2.5 Exponentation test. 55

4.1 Mean square errors for nonnegative x_i 99

6.1 Constants a_{pq} such that $\|x\|_p \le \alpha_{pq}\|x\|_q$, $x \in \mathbb{C}^n$ 121
6.2 Constants a_{pq} such that $\|A\|_p \le \alpha_{pq}\|A\|_q$, $A \in \mathbb{C}^{m\times n}$. 122

7.1 Backward and forward stability. 143

9.1 Times for solution of a linear system of order n 189
9.2 Records for largest dense linear systems solved. 199

11.1 $w_{|A|,|b|}$ values for A = orthog(25). 240
11.2 $w_{|A|,|b|}$ values for A = clement(50) 241
11.3 $w_{|A|,|b|}$ values for A = gfpp(50) 241

12.1 Stability of block and point LU factorization. 256

13.1 Backward errors $\eta_{A,b}(\widehat{x})$ for the ∞-norm. 262
13.2 Mflop rates for inverting a triangular matrix on a Cray 2. . . . 270
13.3 Mflop rates for inverting a full matrix on a Cray 2. 275
13.4 Times (minutes and seconds) for inverting an $n \times n$ matrix. . . 276
13.5 Additional timings for inverting an $n \times n$ matrix. 276
13.6 Gauss-Jordan elimination for Ux =6. 279

16.1 Dates of publication of selected iterative methods. 326

16.2 Result for Jacobi method, $\alpha = \frac{1}{2} - 8^{-j}$. 333
16.3 Results for Jacobi method, a = $-(\frac{1}{2} - 8^{-j})$. 333

19.1 LS backward errors and residual for Vandermonde system. . . . 405

20.1 Backward errors for underdetermined Vandermonde system. . . 422

21.1 Bounds and estimates for $\kappa_\infty(V_n)$. 428
21.2 Parameters in the three-term recurrence (21.6). 433
21.3 Results for dual Chebyshev-Vandermonde-like system. 438

25.1 Results from Cholesky factorization. 496
25.2 Effect of extended run time on Patriot missile operation. 507

26.1 Condition numbers of Hilbert and Pascal matrices. 516

Preface

It has been 30 years since the publication of Wilkinson's books *Rounding Errors in Algebraic Processes* [1088, 1963] and *The Algebraic Eigenvalue Problem* [1089, 1965]. These books provided the first thorough analysis of the effects of rounding errors on numerical algorithms, and they rapidly became highly influential classics in numerical analysis. Although a number of more recent books have included analysis of rounding errors, none has treated the subject in the same depth as Wilkinson.

This book gives a thorough, up-to-date treatment of the behaviour of numerical algorithms in finite precision arithmetic. It combines algorithmic derivations, perturbation theory, and rounding error analysis. Software practicalities are emphasized throughout, with particular reference to LAPACK. The best available error bounds, some of them new, are presented in a unified format with a minimum of jargon. Historical perspective is given to provide insight into the development of the subject, and further information is provided in the many quotations. Perturbation theory is treated in detail, because of its central role in revealing problem sensitivity and providing error bounds. The book is unique in that algorithmic derivations and motivation are given succinctly, and implementation details minimized, so that attention can be concentrated on accuracy and stability results. The book was designed to be a comprehensive reference and contains extensive citations to the research literature.

Although the book's main audience is specialists in numerical analysis, it will be of use to all computational scientists and engineers who are concerned about the accuracy of their results. Much of the book can be understood with only a basic grounding in numerical analysis and linear algebra.

This first two chapters are very general. Chapter 1 describes fundamental concepts of finite precision arithmetic, giving many examples for illustration and dispelling some misconceptions. Chapter 2 gives a thorough treatment of floating point arithmetic and may well be the single most useful chapter in the book. In addition to describing models of floating point arithmetic and the IEEE standard, it explains how to exploit "low-level" features not represented in the models and contains a large set of informative exercises.

In the rest of the book the focus is, inevitably, on numerical linear algebra, because it is in this area that rounding errors are most influential and have been most extensively studied. However, I found that it was impossible to cover the whole of numerical linear algebra in a single volume. The main omission is the area of eigenvalue and singular value computations, which is still the subject of intensive research and requires a book of its own to summarize algorithms, perturbation theory, and error analysis. This book is therefore certainly not a replacement for *The Algebraic Eigenvalue Problem.*

Two reasons why rounding error analysis can be hard to understand are that, first, there is no standard notation and, second, error analyses are often cluttered with re-derivations of standard results. In this book I have used notation that I find nearly always to be the most convenient for error analysis: the key ingredient is the symbol $\gamma_n = nu/(1 - nu)$, explained in §3.1. I have also summarized many basic error analysis results (for example, in Chapters 3 and 8) and made use of them throughout the book. I like to think of these basic results as analogues of the Fortran BLAS (Basic Linear Algebra Subprograms): once available in a standard form they can be used as black boxes and need not be reinvented.

A number of the topics included here have not been treated in depth in previous numerical analysis textbooks. These include floating point summation, block LU factorization, condition number estimation. the Sylvester equation, powers of matrices. finite precision behaviour of stationary iterative methods, Vandermonde systems, and fast matrix multiplication, each of which has its own chapter. But there are also some notable omissions. I would have liked to include a chapter on Toeplitz systems, but this is an area in which stability and accuracy are incompletely understood and where knowledge of the underlying applications is required to guide the investigation. The important problems of updating and downdating matrix factorizations when the matrix undergoes a “small” change have also been omitted due to lack of time and space. A further omission is analysis of parallel algorithms for all the problems considered in the book (though blocked and partitioned algorithms and one particular parallel method for triangular systems are treated). Again, there are relatively few results and this is an area of active research.

Throughout the history of numerical linear algebra, theoretical advances have gone hand in hand with software development. This tradition has continued with LAPACK (1987-), a project to develop a state-of-the-art Fortran package for solving linear equations and eigenvalue problems. LAPACK has enjoyed a synergy with research that has led to a number of important breakthroughs in the design and analysis of algorithms, from the standpoints of both performance and accuracy. A key feature of this book is that it provides the material needed to understand the numerical properties of many of the algorithms in LAPACK, the except ions being the routines for eigenvalue and singular value problems. In particular, the error bounds computed by the LAPACK linear equation solvers are explained, the LAPACK condition estimator is described in detail, and some of the software issues confronted by the LAPACK developers are highlighted. Chapter 25 examines the influence of floating point arithmetic on general numerical software, offering salutary stories, useful techniques, and brief descriptions of relevant codes.

This book has been written with numerical analysis courses in mind, although it is not designed specifically as a textbook. It would be a suitable reference for an advanced course (for example, for a graduate course on Numerical Linear Algebra following the syllabus recommended by the ILAS Education Committee [601, 1993]), and could be used by instructors at all levels as a supplementary text from which to draw examples, historical perspective, statements of results, and exercises. The exercises (actually labelled “problems”) are an important part of the book, and many of them have not, to my knowledge, appeared in textbooks before. Where appropriate I have indicated the source of an exercise; a name without a citation means that the exercise came from private communication or unpublished notes. Research problems given at the end of some sets of exercises emphasize that most of the areas covered are still active.

In addition to surveying and unifying existing results (including some that have not appeared in the mainstream literature) and sometimes improving upon their presentation or proof, this book contains new results. Some of particular note are as follows.

1. The error analysis in §5.3 for evaluation of the Newton interpolating polynomial.
2. The forward error analysis for iterative refinement in §11.1.
3. The error analysis of Gauss-Jordan elimination in §13.4.
4. The unified componentwise error analysis of QR factorization methods in Chapter 18, and the corresponding analysis of their use for solving the least squares problem in Chapter 19.
5. Theorem 20.3, which shows the backward stability of the QR factorization method for computing the minimum 2-norm solution to an underdetermined system.

The Notes and References are an integral part of each chapter. In addition to containing references, historical information, and further details, they include material not covered elsewhere in the chapter, and should always be consulted, in conjunction with the index, to obtain the complete picture.

I have included relatively few numerical examples except in the first chapter. There are two reasons. One is to reduce the length of the book. The second reason is because today it so easy for the reader to perform experiments in MATLAB* or some other interactive system. To this end I have made available the Test Matrix Toolbox, which contains MATLAB M-files for many of the algorithms and special matrices described in the book; see Appendix E.

---

*MATLAB is a registered trademark of The Math Works, Inc.

This book has been designed to be as easy to use as possible. There are thorough name and subject indexes, page headings show chapter and section titles and numbers, and there is extensive cross-referencing. I have adopted the unusual policy of giving with (nearly) every citation not only its numerical location in the bibliography but also the names of the authors and the year of publication. This provides as much information as possible in a citation and reduces the need for the reader to turn to the bibliography.

A BIBTEX database acc-stab-num-alg.bib containing all the references in the bibliography is available over the Internet from the bibnet project (which can be accessed via netlib, described in §C.2).

Special care has been taken to minimize the number of typographical and other errors, but no doubt, some remain. I will be happy to receive notification of errors, as well as comments and suggestions for improvement.

Acknowledgements

Three books, in addition to Wilkinson's, have strongly influenced my research in numerical linear algebra and have provided inspiration for this book: Golub and Van Loan's *Matrix Computations* [470, 1989] (first edition 1983), Parlett's *The Symmetric Eigenvalue Problem* [820, 1980], and Stewart's *Introduction to Matrix Computations* [941, 1973]. Knuth's *The Art of Computer Programming* books [666, 1973-1981] have also influenced my style and presentation.

Jim Demmel has contributed greatly to my understanding of the subject of this book and provided valuable technical help and suggestions. The first two chapters owe much to the work of Velvel Kahan; I am grateful to him for giving me access to unpublished notes and for suggesting improvements to early versions of Chapters 2 and 25. Des Higham read various drafts of the book, offering sound advice and finding improvements that had eluded me.

Other people who have given valuable help, suggestions, or advice are

> Zhaojun Bai, Brad Baxter, Åke Björck, Martin Campbell-Kelly, Shivkumar Chandrasekaran, Alan Edelman, Warren Ferguson, Philip Gill, Gene Golub, George Hall, Sven Hammarling, Andrzej Kielbasiński, Philip Knight, Beresford Parlett, David Silvester, Michael Saunders, Ian Smith, Doron Swade, Nick Trefethen, Jack Williams, and Hongyuan Zha.

David Carlisle provided invaluable help and advice concerning LATEX 2_ε.

Working with SIAM on the publication of this book was a pleasure. Special thanks go to Nancy Abbott (design), Susan Ciambrano (acquisition), Ed Cilurso (production), Beth Gallagher (copy editing), Corey Gray (production),

Mary Rose Muccie (copy editing and indexing), Colleen Robishaw (design), and Sam Young (production).

Research leading to this book has been supported by grants from the Engineering and Physical Sciences Research Council, by a Nuffield Science Research Fellowship from the Nuffield Foundation, and by a NATO Collaborative Research Grant held with J. W. Demmel. I was fortunate to be able to make extensive use of the libraries of the University of Manchester, the University of Dundee, Stanford University, and the University of California, Berkeley.

This book was typeset in LaTeX 2$_\varepsilon$ using the book document style. The references were prepared in BibTeX and the index with MakeIndex. It is difficult to imagine how I could have written the book without these wonderful tools. I used the "big" software from the emTeX distribution, running on a 486DX workstation. I used text editors The Semware Editor (Semware Corporation) and GNU Emacs (Free Software Foundation) and checked spelling with PC-Write (Quicksoft).

Manchester
April 1995

Nicholas J. Higham

About the Dedication

This book is dedicated to the memory of two remarkable English mathematicians, James Hardy Wilkinson (1919-1986), FRS, and Alan Mathison Turing (1912-1954), FRS, both of whom made immense contributions to scientific computation.

Turing's achievements include his paper "On Computable Numbers, with an Application to the Entscheidungsproblem", which answered Hilbert's decidability question using the abstract device now known as a Turing machine [1025, 1936]; his work at Bletchley Park during World War II on breaking the ciphers of the Enigma machine; his 1945 report proposing a design for the Automatic Computing Engine (ACE) at the National Physical Laboratory [1026, 1945]; his 1948 paper on LU factorization and its rounding error analysis [1027, 1948]; his consideration of fundamental questions in artificial intelligence (including his proposal of the "Turing test"); and, during the last part of his life, spent at the University of Manchester, his work on morphogenesis (the development of structure and form in an organism). Turing is remembered through the Turing Award of the Association for Computing Machinery (ACM), which has been awarded yearly since 1966 [3, 1987]. For more about Turing, read the superb biography by Hodges [575, 1983], described by a reviewer as "one of the finest pieces of scholarship to appear in the history of computing" [182, 1984].

Wilkinson, like Turing a Cambridge-trained mathematician, was Turing's assistant at the National Physical Laboratory. When Turing left, Wilkinson managed the group that built the Pilot ACE, contributing to the design and construction of the machine and its software. Subsequently, he used the machine to develop and study a variety of numerical methods. He developed backward error analysis in the 1950s and 1960s publishing the books *Rounding Errors in Algebraic Processes* [1088, 1963][†] (REAP) and *The Algebraic Eigenvalue Problem* [1089, 1965][‡] (AEP), both of which rapidly achieved the status of classics. (AEP was reprinted in paperback in 1988 and, after being out of print for many years, REAP is now also available in paperback.) The AEP was described by the late Professor Leslie Fox as "almost certainly the most important and widely read title in numerical analysis". Wilkinson also contributed greatly to the development of mathematical software. The volume *Handbook for Automatic Computation, Volume* II: *Linear Algebra* [1102, 1971], co-edited with Reinsch, contains high-quality, properly documented software and has strongly influenced subsequent software projects such as the NAG Library, EISPACK, LINPACK, and LAPACK.

[†] REAP has been translated into Polish [1091, 1967] and German [1093, 1969].

[‡] AEP has been translated into Russian [1094, 1970].

Wilkinson received the 1970 Turing Award. In his Turing Award lecture he described life with Turing at the National Physical Laboratory in the 1940s [1096, 1971].

Wilkinson is remembered through SIAM's James H. Wilkinson Prize in Numerical Analysis and Scientific Computing, awarded every 4 years; the Wilkinson Prize for Numerical Software, awarded by Argonne National Laboratory, the National Physical Laboratory, and the Numerical Algorithms Group; and the Wilkinson Fellowship in Scientific Computing at Argonne National Laboratory. For more about Wilkinson see the biographical memoir by Fox [403, 1987], Fox's article [402, 1978], Parlett's essay [821, 1990], the prologue and epilogue of the proceedings [252, 1990] of a conference held in honour of Wilkinson at the National Physical Laboratory in 1987. and the tributes in [23, 1987]. Lists of Wilkinson's publications are given in [403, 1987] and in the special volume of the journal *Linear Algebra and its Applications* (88/89, April 1987) published in his memory.

Chapter 1

Principles of Finite Precision Computation

> *Numerical precision is the very soul of science.*
> -SIR D’ARCY WENTWORTH THOMPSON, *On Growth and Form (1942)*

> *There will always be a small but steady demand for error-analysts to . . .*
> *expose bad algorithms’ big errors and, more important,*
> *supplant bad algorithms with provably good ones.*
> -WILLIAM M. KAHAN, *Interval Arithmetic Options in*
> *the Proposed IEEE Floating Point Arithmetic Standard* (1980)

> *Since none of the numbers which we take out from logarithmic and*
> *trigonometric tables admit of absolute precision,*
> *but are all to a certain extent approximate only,*
> *the results of all calculations performed*
> *by the aid of these numbers can only be approximately true . . .*
> *It may happen, that in* special cases the
> *effect of the errors of the tables is so augmented that*
> *we may be obliged to reject a method,*
> *otherwise the best, and substitute another in its place.*
> -CARL FRIEDRICH GAUSS[1], *Theoria Motus* (1809)

> *Backward error analysis is no panacea;*
> *it may explain errors but not excuse them.*
> -HEWLETT-PACKARD, *HP-15C Advanced Functions Handbook* (1982)

---

[1] Cited in Goldstine [461, 1977, p. 258].

This book is concerned with the effects of finite precision arithmetic on numerical algorithms[2], particularly those in numerical linear algebra. Central to any understanding of high-level algorithms is an appreciation of the basic concepts of finite precision arithmetic. This opening chapter briskly imparts the necessary background material. Various examples are used for illustration, some of them familiar (such as the quadratic equation) but several less well known. Common misconceptions and myths exposed during the chapter are highlighted towards the end, in §1.19.

This chapter has few prerequisites and few assumptions are made about the nature of the finite precision arithmetic (for example, the base, number of digits, or mode of rounding, or even whether it is floating point arithmetic). The second chapter deals in detail with the specifics of floating point arithmetic.

A word of warning: some of the examples from §1.12 onward are special ones chosen to illustrate particular phenomena. You may never see in practice the extremes of behaviour shown here. Let the examples show you what can happen, but do not let them destroy your confidence in finite precision arithmetic!

1.1. Notation and Background

We describe the notation used in the book and briefly set up definitions needed for this chapter.

Generally, we use

capital letters	A, B, C, Δ, Λ	for matrices,
subscripted lower case letters	$\alpha_{ij}, b_{ij}, c_{ij}, \delta_{ij}, \lambda_{ij}$	for matrix elements,
lower case letters	x, y, z, c, g, h	for vectors,
lower case Greek letters	$\alpha, \beta, \gamma, \phi, \rho$	for scalars,

following the widely used convention originally introduced by Householder [587, 1964].

The vector space of all real $m \times n$ matrices is denoted by $\mathbb{R}^{m \times n}$ and the vector space of real n-vectors by $\mathbb{R}^n$. Similarly, $\mathbb{C}^{m \times n}$ denotes the vector space of complex $m \times n$ matrices.

Algorithms are expressed using a pseudocode based on the MATLAB language [232, 1988], [735, 1992]. Comments begin with the % symbol.

Submatrices are specified with the colon notation, as used in MATLAB and Fortran 90: $A(p:q, r:s)$ denotes the submatrix of A formed by the intersection of rows p to q and columns r to s. As a special case, a lone colon as the row or column specifier means to take all entries in that row or column; thus $A(:,j)$ is the jth column of A and $A(i,:)$ the ith row. The values taken by an integer variable are also described using the colon notation: "$i = 1{:}n$" means the same as "$i = 1,2,\dots,n$".

---

[2] For the purposes of this book an algorithm is a MATLAB program; cf. Smale [924, 1990].

Evaluation of an expression in floating point arithmetic is denoted $fl(\cdot)$, and we assume that the basic arithmetic operations op=+,-,*,/ satisfy

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \qquad |\delta| \le u. \tag{1.1}$$

Here, u is the *unit roundoff* (or machine precision), which is typically of order 10^{-8} or 10^{-16} in single and double precision computer arithmetic, respectively, and between 10^{-10} and 10^{-12} on pocket calculators. For more on floating point arithmetic see Chapter 2.

Computed quantities (and, in this chapter only, arbitrary approximations) wear a hat. Thus $\widehat{x}$ denotes the computed approximation to x.

Definitions are often (but not always) indicated by ":=" or "=:", with the colon next to the object being defined.

We make use of the floor and ceiling functions: $\lfloor x \rfloor$ is the largest integer less than or equal to x, and $\lceil x \rceil$ is the smallest integer greater than or equal to x.

The normal distribution with mean μ and variance σ^2 is denoted by $N(\mu,\sigma^2)$.

We measure the cost of algorithms in flops. A *flop* is an elementary floating point operation: +,-,/, or *. We normally state only the highest-order terms of flop counts. Thus, when we say that an algorithm for $n \times n$ matrices requires $2n^3/3$ flops, we really mean $2n^3/3+O(n^2)$ flops.

Other definitions and notation are introduced when needed. Two topics, however, do not fit comfortably into the main text and are described in Appendix B: the singular value decomposition (SVD) and M-matrices.

All our numerical experiments were carried out either in MATLAB 4.2 [735, 1992], sometimes in conjunction with the Symbolic Math Toolbox [204, 1993], or with the Salford Software/Numerical Algorithms Group FTN90[3] Fortran 90 compiler, Version 1.2 [888, 1993]. Whenever we say a computation was "done in Fortran 90" we are referring to the use of this compiler. All the results quoted were obtained on a 486DX workstation, unless otherwise stated, but many of the experiments were repeated on a Sun SPARCstation, using the NAGWare[4] FTN90 compiler [785, 1992]. Both machines use IEEE standard floating point arithmetic and the unit roundoff is $u = 2^{-53} \approx 1.1 \times 10^{-16}$ in MATLAB and in double precision in Fortran 90. (Strictly speaking, in Fortran 90 we should not use the terminology single and double precision but should refer to the appropriate KIND parameters; see, e.g., Metcalf and Reid [749, 1990, §2.6]. However, these terms are vivid and unambiguous in IEEE arithmetic, so we use them throughout the book.)

---

[3] FTN90 is a joint trademark of Salford Software Ltd. and The Numerical Algorithms Group Ltd.

[4] NAGWare is a trademark of The Numerical Algorithms Group Ltd.

1.2. Relative Error and Significant Digits

Let $\hat{x}$ be an approximation to a real number x. The most useful measures of the accuracy of $\hat{x}$ are its *absolute error*

$$E_{\mathrm{abs}}(\hat{x}) = |x - \hat{x}|,$$

and its *relative error*

$$E_{\mathrm{rel}}(\hat{x}) = \frac{|x - \hat{x}|}{|x|}$$

(which is undefined if $x = 0$). An equivalent definition of relative error is $E_{\mathrm{rel}}(\hat{x}) = |r|$, where $\hat{x} = x(1+p)$. Some authors omit the absolute values from these definitions. When the sign is important we will simply talk about "the error $x - \hat{x}$".

In scientific computation, where answers to problems can vary enormously in magnitude, it is usually the relative error that is of interest, because it is scale independent: scaling $x \to \alpha x$ and $\hat{x} \to \alpha\hat{x}$ leaves $E_{\mathrm{rel}}(\hat{x})$ unchanged.

Relative error is connected with the notion of correct significant digits (or correct significant figures). The significant digits in a number are the first nonzero digit and all succeeding digits. Thus 1.7320 has five significant digits, while 0.0491 has only three. What is meant by *correct* significant digits in a number that approximates another seems intuitively clear, but a precise definition is problematic, as we explain in a moment. First, note that for a number $\hat{x}$ with p significant digits there are only $p+1$ possible answers to the question "how many correct significant digits does $\hat{x}$ have?" (assuming $\hat{x}$ is not a constant such as 2.0 that is known exactly). Therefore the number of correct significant digits is a fairly crude measure of accuracy in comparison with the relative error. For example, in the following two cases $\hat{x}$ agrees with x to three but not four significant digits by any reasonable definition, yet the relative errors differ by a factor of about 44:

$$x = 1.00000, \quad \hat{x} = 1.00499, \quad E_{\mathrm{rel}}(\hat{x}) = 4.99 \times 10^{-3},$$
$$x = 9.00000, \quad \hat{x} = 8.99899, \quad E_{\mathrm{rel}}(\hat{x}) = 1.12 \times 10^{-4}.$$

Here is a possible definition of correct significant digits: *an approximation $\hat{x}$ to x has p correct significant digits if $\hat{x}$ and x round to the same number to p significant digits. Rounding* is the act of replacing a given number by the nearest p significant digit number, with some rule for breaking ties when there are two nearest. This definition of correct significant digits is mathematically elegant and agrees with intuition most of the time. But consider the numbers

$$x = 0.9949, \quad \hat{x} = 0.9951.$$

According to the definition $\hat{x}$ does not have two correct significant digits ($x \to 0.99$, $\hat{x} \to 1.0$), but does have one and three correct significant digits!

or, if the data is itself the solution to another problem, it may be the result of errors in an earlier computation. The effects of errors in the data are generally easier to understand than the effects of rounding errors committed during a computation, because data errors can be analysed using perturbation theory for the problem at hand, while intermediate rounding errors require an analysis specific to the given method. This book contains perturbation theory for most of the problems considered, for example, in Chapters 7 (linear systems), 19 (the least squares problem), and 20 (underdetermined systems).

Analysing truncation errors, or discretization errors, is one of the major tasks of the numerical analyst. Many standard numerical methods (for example, the trapezium rule for quadrature, Euler's method for differential equations, and Newton's method for nonlinear equations) can be derived by taking finitely many terms of a Taylor series. The terms omitted constitute the truncation error, and for many methods the size of this error depends on a parameter (often called *h*, "the stepsize") whose appropriate value is a compromise between obtaining a small error and a fast computation.

Because the emphasis of this book is on finite precision computation, with virtually no mention of truncation errors, it would be easy for the reader to gain the impression that the study of numerical methods is dominated by the study of rounding errors. This is certainly not the case. Trefethen explains it well when he discusses how to define numerical analysis [1016, 1992]:

> Rounding errors and instability are important, and numerical analysts will always be the experts in these subjects and at pains to ensure that the unwary are not tripped up by them. But our central mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.

In this quotation "uncomputable" means that approximations are necessary, and thus Trefethen's point is that developing good approximations is a more fundamental task than analysing the effects of rounding errors on those approximations.

A possible way to avoid rounding and truncation errors (but not data errors) is to try to solve a problem using a symbolic manipulation package, such as Maple[5] [199, 1991] or Mathematica[6] [1109, 1991]. Indeed, we have used this approach to compute "exact answers" in some of our numerical experiments, While we acknowledge the value of symbolic manipulation as part of the toolkit of the scientific problem solver, we do not study it in this book.

---

[5] Maple is a registered trademark of Waterloo Maple Software.

[6] Mathematica is a registered trademark of Wolfram Research Inc.

A definition of correct significant digits that does not suffer from the latter anomaly states that $\widehat{x}$ *agrees with* x *to p significant digits if* $|x-\widehat{x}|$ *is less than half a unit in the pth significant digit of* x.. However, this definition implies that 0.123 and 0.127 agree to two significant digits, whereas many people would say that they agree to less than two significant digits.

In summary, while the number of correct significant digits provides a useful way in which to think about the accuracy of an approximation, the relative error is a more precise measure (and is base independent). Whenever we give an approximate answer to a problem we should aim to state an estimate or bound for the relative error.

When x and $\widehat{x}$ are vectors the relative error is most often defined with a norm, as $\|x-\widehat{x}\|/\|x\|$. For the commonly used norms $\|x\|_\infty := \max_i |x_i|$, $\|x\|_1 := \sum_i |x_i|$, and $\|x\|_2 := (x^Tx)^{1/2}$, the inequality $\|x-\widehat{x}\|/\|x\| \le \frac{1}{2} \times 10^{-p}$ implies that components x_i with $|x_i| \approx \|x\|$ have about p correct significant decimal digits, but for the smaller components the inequality merely bounds the absolute error.

A relative error that puts the individual relative errors on an equal footing is the *componentwise relative error*

$$\max_i \frac{|x_i-\widehat{x}_i|}{|x_i|},$$

which is widely used in error analysis and perturbation analysis (see Chapter 7, for example).

As an interesting aside we mention the “tablemaker’s dilemma”. Suppose you are tabulating the values of a transcendental function such as the sine function and a particular entry is evaluated as 0.124|500000000 correct to a few digits in the last place shown, where the vertical bar follows the final significant digit to be tabulated. Should the final significant digit be 4 or 5? The answer depends on whether there is a nonzero trailing digit and, in principle, we may never be able answer the question by computing only a finite number of digits.

1.3. Sources of Errors

There are three main sources of errors in numerical computation: rounding, data uncertainty, and truncation.

Rounding errors, which are an unavoidable consequence of working in finite precision arithmetic, are largely what this book is about. The remainder of this chapter gives basic insight into rounding errors and their effects.

Uncertainty in the data is always a possibility when we are solving practical problems. It may arise in several ways: from errors in measuring physical quantities, from errors in storing the data on the computer (rounding errors),

1.4. Precision Versus Accuracy

The terms accuracy and precision are often confused or used interchangeably, but it is worth making a distinction between them. *Accuracy* refers to the absolute or relative error of an approximate quantity. *Precision* is the accuracy with which the basic arithmetic operations +,-,*,/ are performed, and for floating point arithmetic is measured by the unit roundoff u (see (1.1)). Accuracy and precision are the same for the scalar computation $c = a*b$, but accuracy can be much worse than precision in the solution of a linear system of equations, for example.

It is important to realize that *accuracy is not limited by precision,* at least in theory. This may seem surprising, and may even appear to contradict many of the results in this book. However, arithmetic of a given precision can be used to simulate arithmetic of arbitrarily high precision, as explained in §25.9. (The catch is that such simulation is too expensive to be of practical use for routine computation.) In all our error analyses there is an implicit assumption that the given arithmetic is not being used to simulate arithmetic of a higher precision.

1.5. Backward and Forward Errors

Suppose that an approximation $\widehat{y}$ to $y = f(x)$ is computed in an arithmetic of precision u, where f is a real scalar function of a real scalar variable. How should we measure the “quality” of ?

In most computations we would be happy with a tiny relative error, $E_{\mathrm{rel}}(\widehat{y}) \approx u$, but this cannot always be achieved. Instead of focusing on the relative error of $\widehat{y}$ we can ask “for what set of data have we actually solved our problem?“, that is, for what Δx do we have $\widehat{y} = f(x + \Delta x)$? In general, there may be many such Δx, so we should ask for the smallest one. The value of $|\Delta x|$ (or min $|\Delta x|$), possibly divided by $|x|$, is called the *backward error.* The absolute and relative errors of $\widehat{y}$ are called *forward errors,* to distinguish them from the backward error. Figure 1.1 illustrates these concepts.

The process of bounding the backward error of a computed solution is called *backward error analysis,* and its motivation is twofold. First, it interprets rounding errors as being equivalent to perturbations in the data. The data frequently contains uncertainties due to previous computations or errors committed in storing numbers on the computer. If the backward error is no larger than these uncertainties then the computed solution can hardly be criticized-it may be the solution we are seeking, for all we know. The second attraction of backward error analysis is that it reduces the question of bounding or estimating the forward error to perturbation theory, which for many problems is well understood (and only has to be developed once, for the

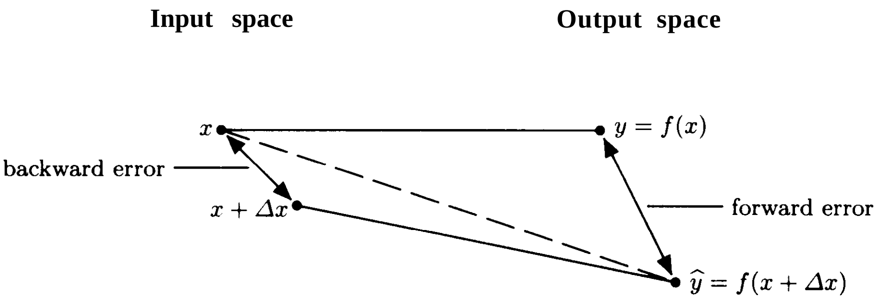


Figure 1.1. *Backward and forward errors for* $y = f(x)$. *Solid line = exact; dotted line = computed.*

given problem, and not for each method). We discuss perturbation theory in the next section.

A method for computing $y = f(x)$ is called *backward stable* if, for any it produces a computed $\widehat{y}$ with a small backward error, that is, $\widehat{y} = f(x + \Delta x)$ for some small Δx. The definition of "small" will be context dependent. In general, a given problem has several methods of solution, some of which are backward stable and some not.

As an example, assumption (1.1) says that the computed result of the operation $x \pm y$ is the exact result for perturbed data $x(1 + \delta)$ and $y(1 + \delta)$ with $|\delta| \le u$; thus addition and subtraction are, by assumption, backward stable operations.

Most routines for computing the cosine function do not satisfy $\widehat{y} = \cos(x + \Delta x)$ with a relatively small Δx, but only the weaker relation $\widehat{y} + \Delta y = \cos(x + \Delta x)$, with relatively small Δy and Δx. A result of the form

$$\widehat{y} + \Delta y = f(x + \Delta x), \qquad |\Delta y| \le \epsilon |y|, \quad |\Delta x| \le \eta |x| \tag{1.2}$$

is known *as* a *mixed forward-backward* error result and is illustrated in Figure 1.2. Provided that ϵ and η are sufficiently small, (1.2) says that *the computed value* $\widehat{y}$ *scarcely differs from the value* $\widehat{y} + \Delta y$ *that would have been produced by an input* $x + \Delta x$ *scarcely different from the actual input* x. Even more simply, $\widehat{y}$ is almost the right answer for almost the right data.

In general, an algorithm is called *numerically stable* if it is stable in the mixed forward-backward error sense of (1.2) (hence a backward stable algorithm can certainly be called numerically stable). Note that this definition is specific to problems where rounding errors are the dominant form of errors. The term *stability* has different meanings in other areas of numerical analysis.

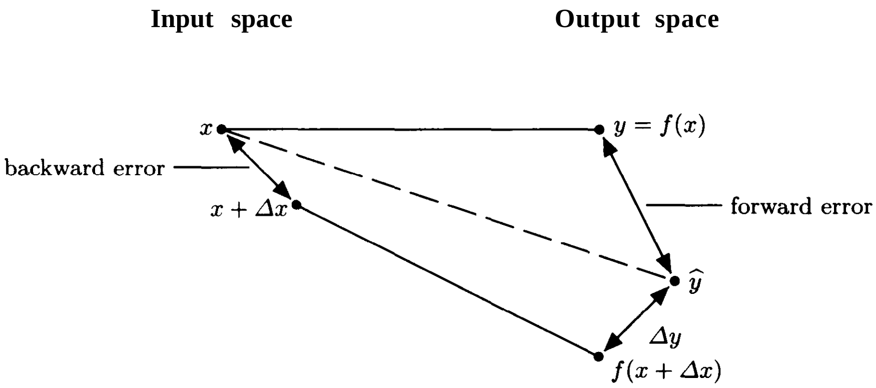


Figure 1.2. *Mixed forward-backward error for* $y = f(x)$. *Solid line = exact; dotted line = computed.*

1.6. Conditioning

The relationship between forward and backward error for a problem is governed by the conditioning of the problem, that is, the sensitivity of the solution to perturbations in the data. Continuing the $y = f(x)$ example of the previous section, let an approximate solution $\widehat{y}$ satisfy $\widehat{y} = f(x + \Delta x)$. Then, assuming for simplicity that f is twice continuously differentiable,

$$\widehat{y} - y = f(x + \Delta x) - f(x) = f'(x)\Delta x + \frac{f''(x + \theta\Delta x)}{2!}(\Delta x)^2, \quad \theta \in (0,1),$$

and we can bound or estimate the right-hand side. This expansion leads to the notion of condition number. Since

$$\frac{\widehat{y} - y}{y} = \left(\frac{xf'(x)}{f(x)}\right)\frac{\Delta x}{x} + O\big((\Delta x)^2\big),$$

the quantity

$$c(x) = \left|\frac{xf'(x)}{f(x)}\right|$$

measures, for small Δx, the relative change in the output for a given relative change in the input, and it is called the *(relative) condition number* of f. If x or f is a vector then the condition number is defined in a similar way using norms and it measures the *maximum* relative change, which is attained for some, but not all, vectors Δx.

As an example, consider the function $f(x) = \log x$. The condition number is $(x) = |1/\log x|$, which is large for $x \approx 1$. This means that a small relative change in x can produce a much larger relative change in $\log x$ for $x \approx 1$. The reason is that a small relative change in produces a small *absolute* change in $f(x) = \log x$ (since $f(x+\Delta x) \approx f(x) + f'(x)\Delta x = f(x) + \Delta x/x$) and that change in $\log x$ may be large in a relative sense.

When backward error, forward error, and the condition number are defined in a consistent fashion we have the useful rule of thumb that

forward error condition number × backward error,

with approximate equality possible. One way to interpret this rule of thumb is to say that *the computed solution to an ill-conditioned problem can have a large forward error.* For even if the computed solution has a small backward error, this error can be amplified by a factor as large as the condition number when passing to the forward error.

One further definition is useful. If a method produces answers with forward errors of similar magnitude to those produced by a backward stable method, then it is called *forward stable.* Such a method need not be backward stable itself. Backward stability implies forward stability, but not vice versa. An example of a method that is forward stable but not backward stable is Cramer's rule for solving a 2 × 2 linear system, which is discussed in §1.10.1.

1.7. Cancellation

Cancellation is what happens when two nearly equal numbers are subtracted. It is often, but not always, a bad thing. Consider the function $f(x) = (1 - \cos x)/x^2$. With $x = 1.2\times 10^{-5}$ the value of $\cos x$ rounded to 10 significant figures is

$$c = 0.9999\ 9999\ 99,$$

so that

$$1 - c = 0.0000\ 0000\ 01.$$

Then $(1 - c)/x^2 = 10^{-10}/1.44\times 10^{-10} = 0.6944\ldots$, which is clearly wrong given the fact that $0 \le f(x) \le 1/2$ for all $x \ne 0$. A 10 significant figure approximation to $\cos x$ is therefore not sufficient to yield a value of $f(x)$ with even one correct figure. The problem is that $1 - c$ has only 1 significant figure. The subtraction $1 - c$ is *exact,* but this subtraction produces a result of the same size as the error in c. In other words, the subtraction elevates the importance of the earlier error. In this particular example it is easy to rewrite $f(x)$ to avoid the cancellation. Since $\cos x = 1 - 2\sin^2(x/2)$,

$$f(x) = \frac{1}{2}\left(\frac{\sin(x/2)}{x/2}\right)^2.$$

Evaluating this second formula for $f(x)$ with a 10 significant figure approximation to $\sin(x/2)$ yields $f(x) = 0.5$, which is correct to 10 significant figures.

To gain more insight into the cancellation phenomenon consider the subtraction (in exact arithmetic) $\widehat{x} = \widehat{a} - b$, where $\widehat{a} = a(1+\Delta a)$ and $\widehat{b} = b(1+\Delta b)$. The terms *Da* and *Db* are relative errors or uncertainties in the data, perhaps attributable to previous computations. With $x = a - b$ *we* have

$$\left|\frac{x-\widehat{x}}{x}\right| = \left|\frac{-a\Delta a - b\Delta b}{a-b}\right| \le \max(|\Delta a|, |\Delta b|)\frac{|a|+|b|}{|a-b|}.$$

The relative error bound for $\widehat{x}$ is large when $|a - b| << |a| + |b|$, that is, when there is heavy cancellation in the subtraction. This analysis shows that subtractive cancellation causes relative errors or uncertainties already present in $\widehat{a}$ and b to be magnified. In other words, subtractive cancellation *brings earlier errors into prominence.*

It is important to realize that cancellation is not always a bad thing. There are several reasons. First, the numbers being subtracted may be error free, as when they are from initial data that is known exactly. The computation of divided differences, for example, involves many subtractions, but half of them involve the initial data and are harmless for suitable orderings of the points (see §5.3 and §21.3). The second reason is that cancellation may be a symptom of intrinsic ill conditioning of a problem, and may therefore be unavoidable. Third, the effect of cancellation depends on the role that the result plays in the remaining computation. For example, if $x >> y \approx z > 0$ then the cancellation in the evaluation of $x + (y - z)$ is harmless.

1.8. Solving a Quadratic Equation

Mathematically, the problem of solving the (real) quadratic equation $ax^2 + bx + c = 0$ is trivial: there are two roots (if $a \neq 0$)) given by

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{1.3}$$

Numerically, the problem is more challenging, as neither the successful evaluation of (1.3) nor the accuracy of the computed roots can be taken for granted.

The easiest issue to deal with is the choice of formula for computing the roots. If $b^2 >> |4ac|$ then $\sqrt{b^2 - 4ac} \approx |b|$, and so for one choice of sign the formula (1.3) suffers massive cancellation. This is damaging cancellation because one of the arguments, $(\sqrt{b^2 - 4ac})$, is inexact, so the subtraction brings into prominence the earlier rounding errors. How to avoid the cancellation is well known: obtain the larger root (in absolute value), x_1, from

$$x_1 = \frac{-\left(b + \operatorname{sign}(b)\sqrt{b^2 - 4ac}\right)}{2a},$$

and the other from the equation $x_1 x_2 = c/a$.

Unfortunately, there is a more pernicious source of cancellation: the subtraction $b^2 - 4ac$. Accuracy is lost here when $b^2 \approx 4ac$ (the case of nearly equal roots), and no algebraic rearrangement can avoid the cancellation. The only way to guarantee accurate computed roots is to use double precision (or some trick tantamount to the use of double precision) in the evaluation of $b^2 - 4ac$.

Another potential difficulty is underflow and overflow. If we apply the formula (1.3) in IEEE single precision arithmetic (described in §2.3) to the equation $10^{20}x^2 - 3\cdot10^{20}x+2\cdot10^{20} = 0$ then overflow occurs, since the maximum floating point number is of order 10 the roots, however, are innocuous: $x = 1$ and $x = 2$. Dividing through the equation by $\max(|a|, |b|, |c|) = 10^{20}$ cures the problem, but this strategy is ineffective for the equation $10^{-20}x^2 - 3x+2\cdot10^{20} = 0$, whose roots are 10^{20} and $2\cdot10^{20}$. In the latter equation we need to scale the variable: defining $x = 10^{20}y$ gives $10^{20}y^2-3\cdot10^{20}y+2\cdot10^{20}=0$, which is the first equation we considered. These ideas can be built into a general scaling strategy (see the Notes and References), but the details are nontrivial.

As this discussion indicates, not only is it difficult to devise an accurate and robust algorithm for solving a quadratic equation, but it is a nontrivial task to prepare specifications that define precisely what “accurate” and “robust” mean for a given system of floating point arithmetic.

1.9. Computing the Sample Variance

In statistics the sample variance of n numbers $x_1, \ldots, x_n$ is defined as

$$s_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2, \tag{1.4}$$

where the sample mean

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Computing s_n^2 from this formula requires two passes through the data, one to compute $\overline{x}$ and the other to accumulate the sum of squares. A two-pass computation is undesirable for large data sets or when the sample variance is to be computed as the data is generated. An alternative formula, found in many statistics textbooks, uses about the same number of operations but requires only one pass through the data:

$$s_n^2 = \frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}\left(\sum_{i=1}^{n} x_i\right)^2\right). \tag{1.5}$$

This formula is very poor in the presence of rounding errors because it computes the sample variance as the difference of two positive numbers, and therefore can suffer severe cancellation that leaves the computed answer dominated by roundoff. In fact, the computed answer can be negative, an event aptly described by Chan, Golub, and LeVeque [194, 1983] as "a blessing in disguise since this at least alerts the programmer that disastrous cancellation has occurred". In contrast, the original formula (1.4) always yields a very accurate (and nonnegative) answer, unless n is large (see Problem 1.10). Surprisingly, current calculators from more than one manufacturer (but not Hewlett-Packard) appear to use the one-pass formula, and they list it in their manuals.

As an example, if $x = [10000, 10001, 10002]^T$ then, in single precision arithmetic ($u \approx 6 \times 10^{-8}$), the sample variance is computed as 1.0 by the two-pass formula (relative error 0) but 0.0 by the one-pass formula (relative error 1). It might be argued that this data should be shifted by some estimate of the mean before applying the one-pass formula ($x_i \to x_i - d$, $i = 1\colon n$, which does not change s_n^2), but a good estimate is not always available and there are alternative one-pass algorithms that will always produce an acceptably accurate answer. For example, instead of accumulating $\sum_i x_i$ and $\sum_i x_i^2$ we can accumulate

$$M_k := \frac{1}{k}\sum_{i=1}^{k} x_i \quad \text{and} \quad Q_k := \sum_{i=1}^{k}(x_i - M_k)^2 = \sum_{i=1}^{k} x_i^2 - \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right)^2,$$

which can be done via the updating formulae

$$M_1 = x_1, \qquad M_k = M_{k-1} + \frac{x_k - M_{k-1}}{k}, \qquad k = 2\colon n, \tag{1.6a}$$

$$Q_1 = 0, \qquad Q_k = Q_{k-1} + \frac{(k-1)(x_k - M_{k-1})^2}{k}, \qquad k = 2\colon n, \tag{1.6b}$$

after which $s_n^2 = Q_n/(n-1)$. Note that the only subtractions in these recurrences are relatively harmless ones that involve the data x_i. For the numerical example above, (1.6) produces the exact answer. The updating formulae (1.6) are numerically stable, though their error bound is not as small as the one for the two-pass formula (it is proportional to the condition number κ_N in Problem 1.7).

The problem of computing the sample variance illustrates well how mathematically equivalent formulae can have different numerical stability properties.

1.10. Solving Linear Equations

For an approximate solution y to a linear system $Ax = b$ ($A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$) the forward error is defined as $\|x-y\|/\|x\|$, for some appropriate norm.

Another measure of the quality of y, more or less important depending on the circumstances, is the size of the residual $r = b - Ay$. When the linear system comes from an interpolation problem, for example, we are probably more interested in how closely Ay represents b than in the accuracy of y. The residual is scale dependent: multiply A and b by a and r is multiplied by a. One way to obtain a scale-independent quantity is to divide by $\|A\|\,\|y\|$, yielding the *relative residual*

$$\rho(y) := \frac{\|b - Ay\|}{\|A\|\,\|y\|}.$$

The importance of the relative residual is explained by the following result, which was probably first proved by Wilkinson (see the Notes and References). We use the 2-norm, defined by $\|x\|_2 = (x^T x)^{1/2}$ and $\|A\|_2 = \max_{x \neq 0} \|Ax\|_2 / \|x\|_2$.

Lemma 1.1. *With the notation above, and for the 2-norm,*

$$\rho(y) = \min\left\{ \frac{\|\Delta A\|_2}{\|A\|_2} : (A + \Delta A)y = b \right\}.$$

Proof. If $(A+\Delta A)y = b$ then $r := b-Ay = DAy$, so $\|r\|_2 < \|\Delta A\|_2 \|y\|_2$, giving

$$\frac{\|\Delta A\|_2}{\|A\|_2} \geq \frac{\|r\|_2}{\|A\|_2 \|y\|_2} = \rho(y). \tag{1.7}$$

On the other hand, $(A+\Delta A)y = b$ for $DA = ry^T/(y^T y)$ and $\|\Delta A\|_2 = \|r\|_2/\|y\|_2$, so the bound (1.7) is attainable. □

Lemma 1.1 says that $p(y)$ measures how much A (but not b) must be perturbed in order for y to be the exact solution to the perturbed system, that is, $p(y)$ equals a normwise relative backward error. If the data A and b are uncertain and $p(y)$ is no larger than this uncertainty (e.g., $p(y) = O(u)$) then the approximate solution y must be regarded as very satisfactory. For other problems the backward error may not be as easy to compute as it is for a general linear system, as we will see for the Sylvester equation (§15.2) and the least squares problem (§19.7).

To illustrate these concepts we consider two specific linear equation solvers: Gaussian elimination with partial pivoting (GEPP) and Cramer's rule.

1.10.1. GEPP Versus Cramer's Rule

Cramer's rule says that the components of the solution to a linear system $Ax = b$ are given by $x_i = \det(A_i(b))/\det(A)$, where $A_i(b)$ denotes A with its ith column replaced by b. These formulae are a prime example of a method that is mathematically elegant, but useless for solving practical problems. The two flaws in Cramer's rule are its computational expense and its numerical instability. The computational expense needs little comment, and is, fortunately, explained in most modern linear algebra textbooks (for example, Strang [961, 1993] cautions the student "it would be crazy to solve equations that way"). The numerical instability is less well known, but not surprising. It is present even for $n = 2$, as a numerical example shows.

We formed a 2 × 2 system $Ax = b$ with condition number $\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 \approx 10^{13}$, and solved the system by both Cramer's rule and GEPP in MATLAB (unit roundoff $u \approx 1.1 \times 10^{-16}$). The results were as follows, where $r = b - A\widehat{x}$:

Cramer's rule		GEPP	
$\widehat{x}$	$r/(\|A\|_2\|\widehat{x}\|_2)$	$\widehat{x}$	$r/(\|A\|_2\|\widehat{x}\|_2)$
1.0000	1.5075×10^{-7}	1.0002	-4.5689×10^{-17}
2.0001	1.9285×10^{-7}	2.0004	-2.1931×10^{-17}

The scaled residual for GEPP is pleasantly small-of order the unit roundoff. That for Cramer's rule is ten orders of magnitude larger, showing that the computed solution $\widehat{x}$ from Cramer's rule does not closely satisfy the equations, or, equivalently, does not solve a nearby system. The solutions themselves are similar, both being accurate to three significant figures in each component but incorrect in the fourth significant figure. This is the accuracy we would expect from GEPP because of the rule of thumb "forward error $\lesssim$ backward error × condition number". That Cramer's rule is as accurate as GEPP in this example, despite its large residual, is perhaps surprising, but it is explained by the fact that Cramer's rule is forward stable for $n = 2$; see Problem 1.9. For general n, the accuracy and stability of Cramer's rule depend on the method used to evaluate the determinants, and satisfactory bounds are not known even for the case where the determinants are evaluated by GEPP.

The small residual produced by GEPP in this example is typical: error analysis shows that GEPP is guaranteed to produce a relative residual of order u when $n = 2$ (see §9.2). To see how remarkable a property this is, consider the rounded version of the exact solution: $z = fl(x) = x + \Delta x$, where $\|\Delta x\|_2 \le u\|x\|_2$. The residual of z satisfies $\|b - Az\|_2 = \|-A\Delta x\|_2 \le u\|A\|_2\|x\|_2 \approx u\|A\|_2\|z\|_2$. Thus *the computed solution from GEPP has about as small a residual as the rounded exact solution, irrespective of its accuracy.*

Expressed another way, the errors in GEPP are highly correlated so as to produce a small residual. To emphasize this point, the vector [1.0006,2.0012], which agrees with the exact solution of the above problem to five significant figures (and therefore is more accurate than the solution produced by GEPP), has a relative residual $\|r\|_2/(\|A\|_2\|\widehat{x}\|_2)$ of order 10^{-6}.

Table 1.1. *Computed approximations* $\widehat{f}_n = fl((1+1/n)^n)$ *to* $e = 2.71828\ldots$

n	$\widehat{f}_n$	$\lvert e - \widehat{f}_n \rvert$
10^1	2.593743	1.25×10^{-1}
10^2	2.704811	1.35×10^{-2}
10^3	2.717051	1.23×10^{-3}
10^4	2.718597	3.15×10^{-4}
10^5	2.721962	3.68×10^{-3}
10^6	2.595227	1.23×10^{-1}
10^7	3.293968	5.76×10^{-1}

1.11. Accumulation of Rounding Errors

Since the first electronic computers were developed in the 1940s, comments along the following lines have often been made: “The enormous speed of current machines means that in a typical problem many millions of floating point operations are performed. This in turn means that rounding errors can potentially accumulate in a disastrous way.” This sentiment is true, but misleading. Most often, instability is caused not by the accumulation of millions of rounding errors. but by the insidious growth of just a few rounding errors.

As an example, let us approximate $e = \exp(1)$ by taking finite n in the definition $e := \lim_{n\to\infty}(1 + 1/n)^n$. Table 1.1 gives results computed in Fortran 90 in single precision ($u \approx 6 \times 10^{-8}$).

The approximations are poor, degrading as n approaches the reciprocal of the machine precision. For n a power of 10, l/n has a nonterminating binary expansion. When $1+1/n$ is formed for n a large power of 10, only a few significant digits from l/n are retained in the sum. The subsequent exponentiation to the power n, even if done exactly, must produce an inaccurate approximation to e(indeed, doing the exponentiation in double precision does not change any of the numbers shown in Table 1.1). Therefore a single rounding error is responsible for the poor results in Table 1.1.

There is a way to compute $(1+1/n)^n$ more accurately, using only single precision arithmetic; it is the subject of Problem 1.5.

Strassen’s method for fast matrix multiplication provides another example of the unpredictable relation between the number of arithmetic operations and the error. If we evaluate $fl(AB)$ by Strassen’s method, for $n\times n$ matrices A and B, and we look at the error as a function of the recursion threshold $n_0 \leq n$. we find that while the number of operations decreases as n_0 decreases from n to 8, the error typically *increases;* see §22.2.2.

1.12. Instability Without Cancellation

It is tempting to assume that calculations free from subtractive cancellation must be accurate and stable, especially if they involve only a small number of operations. The three examples in this section show the fallacy of this assumption.

1.12.1. The Need for Pivoting

Suppose we wish to compute an LU factorization

$$A = \begin{bmatrix} \epsilon & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}, \qquad 0 < \epsilon \ll 1.$$

Clearly, $u_{11} = \epsilon$, $u_{12} = -1$, $l_{21} = \epsilon^{-1}$, and $u_{22} = 1 - l_{21}u_{12} = 1 + \epsilon^{-1}$. In floating point arithmetic, if ϵ is sufficiently small then $\widehat{u}_{22} = fl(1 + \epsilon^{-1})$ evaluates to ϵ^{-1}. Assuming l_{21} is computed exactly, we then have

$$A - \widehat{L}\widehat{U} = \begin{bmatrix} \epsilon & -1 \\ 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{bmatrix} \begin{bmatrix} \epsilon & -1 \\ 0 & \epsilon^{-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus the computed LU factors fail completely to reproduce A. Notice that there is no subtraction in the formation of L and U. Furthermore, the matrix A is very well conditioned ($\kappa_\infty(A) = 4/(1+\epsilon)$). The problem, of course, is with the choice of ϵ as the pivot. The partial pivoting strategy would interchange the two rows of A before factorizing it, resulting in a stable factorization.

1.12.2. An Innocuous Calculation?

For any $x \geq 0$ the following computation leaves x unchanged:

```
for $i = 1:60$
    $x = \sqrt{x}$
end
for $i = 1:60$
    $x = x^2$
end
```

Since the computation involves no subtractions and all the intermediate numbers lie between 1 and x, we might expect it to return an accurate approximation to x in floating point arithmetic.

On the HP 48G calculator, starting with $x = 100$ the algorithm produces $x = 1.0$. In fact, for any x, the calculator computes, in place of $f(x) = x$, the function

$$\widehat{f}(x) = \begin{cases} 0, & 0 \leq x < 1, \\ 1, & x \geq 1. \end{cases}$$

The calculator is producing a completely inaccurate approximation to $f(x)$ in just 120 operations on nonnegative numbers. How can this happen?

The positive numbers x representable on the HP 48G satisfy $10^{-499} \le x \le 9.999\ldots \times 10^{499}$. If we define $r(x) = x^{1/2^{60}}$ then, for any machine number $x \ge 1$,

$$\begin{aligned} 1 \le r(x) < r(10^{500}) &= 10^{500/2^{60}} \\ &= e^{500\cdot 2^{-60}\cdot \log 10} < e^{10^{-15}} \\ &= 1 + 10^{-15} + \tfrac{1}{2}\cdot 10^{-30} + \cdots, \end{aligned}$$

which rounds to 1, since the HP 48G works to about 12 decimal digits. Thus for $x > 1$, the repeated square roots reduce x to 1.0, which the squarings leave unchanged.

For $0 < x < 1$ we have

$$x \le 0.\underbrace{99\ldots9}_{12}$$

on a 12-digit calculator, so we would expect the square root to satisfy

$$\begin{aligned} \sqrt{x} &\le (1 - 10^{-12})^{1/2} = 1 - \tfrac{1}{2}\cdot 10^{-12} - \tfrac{1}{8}\cdot 10^{-24} - \cdots \\ &= 0.\underbrace{99\ldots9}_{12}4\underbrace{99\ldots9}_{11}87499\ldots \end{aligned}$$

This upper bound rounds to the 12 significant digit number $0.99\ldots9$. Hence after the 60 square roots we have on the calculator a number $x \le 0.99\ldots9$. The 60 squarings are represented by $s(x) = x^{2^{60}}$, and

$$\begin{aligned} s(x) \le s(0.99\ldots9) &= (1 - 10^{-12})^{2^{60}} \\ &= 10^{2^{60}\log(1-10^{-12})\log_{10} e} \\ &\approx 10^{-2^{60}\cdot 10^{-12}\cdot \log_{10} e} \\ &\approx 3.6 \times 10^{-500708}. \end{aligned}$$

Because it is smaller than the smallest positive representable number, this result is set to zero on the calculator--a process known as *underflow*. (The converse situation, in which a result exceeds the largest representable number, is called *overflow*.)

The conclusion is that there is nothing wrong with the calculator. This innocuous-looking calculation simply exhausts the precision and range of a machine with 12 digits of precision and a 3-digit exponent.

1.12.3. An Infinite Sum

It is well known that $\sum_{k=1}^{\infty} k^{-2} = \pi^2/6 = 1.6449\ 3406\ 6848\ldots$. Suppose we were not aware of this identity and wished to approximate the sum numerically. The most obvious strategy is to evaluate the sum for increasing k until the computed sum does not change. In Fortran 90 in single precision this yields the value 1.6447 2532, which is first attained at $k = 4096$. This agrees with the exact infinite sum to just four significant digits out of a possible nine.

The explanation for the poor accuracy is that we are summing the numbers from largest to smallest, and the small numbers are unable to contribute to the sum. For $k = 4096$ we are forming $s + 4096^{-2} = s + 2^{-24}$, where $s \approx 1.6$. Single precision corresponds to a 24-bit mantissa, so the term we are adding to s "drops off the end" of the computer word, as do all successive terms.

The simplest cure for this inaccuracy is to sum in the opposite order: from smallest to largest. Unfortunately, this requires knowledge of how many terms to take before the summation begins. With 10^9 terms we obtain the computed sum 1.6449 3406, which is correct to eight significant digits.

For much more on summation, see Chapter 4.

1.13. Increasing the Precision

When the only source of errors is rounding, a common technique for estimating the accuracy of an answer is to recompute it at a higher precision and to see how many digits of the original and the (presumably) more accurate answer agree. We would intuitively expect any desired accuracy to be achievable by computing at a high enough precision. This is certainly the case for algorithms possessing an error bound proportional to the precision, which includes all the algorithms described in the subsequent chapters of this book. However, since an error bound is not necessarily attained, there is no guarantee that a result computed in t digit precision will be more accurate than one computed in s digit precision, for a given $t > s$; in particular, for a very ill conditioned problem both results could have no correct digits.

For illustration, consider the system $Ax = b$, where A is the inverse of the 5×5 Hilbert matrix and $b_i = (-1)^i i$. (For details of the matrices used in this experiment see Chapter 26.) We solved the system in varying precisions with unit roundoffs $u = 2^{-t}$, $t = 15{:}40$, corresponding to about 4 to 12 decimal places of accuracy. (This was accomplished in MATLAB by using the function chop from the Test Matrix Toolbox to round the result of every arithmetic operation to t bits; see Appendix E.) The algorithm used was Gaussian elimination (without pivoting), which is perfectly stable for this symmetric positive definite matrix. The upper plot of Figure 1.3 shows t against the relative errors $\|x-\widehat{x}\|_\infty/\|x\|_\infty$ and the relative residuals $\|b - A\widehat{x}\|_\infty/(\|A\|_\infty\|\widehat{x}\|_\infty)$. The lower plot of Figure 1.3 gives corresponding results for $A = P_5 + 5I$, where P_5 is the Pascal matrix of order 5. The condition numbers $\kappa_\infty(A)$ are 1.62×10^2 for the inverse Hilbert matrix and 9.55×10^5 for the shifted Pascal matrix. In both cases the general trend is that increasing the precision decreases the residual and relative error, but the behaviour is

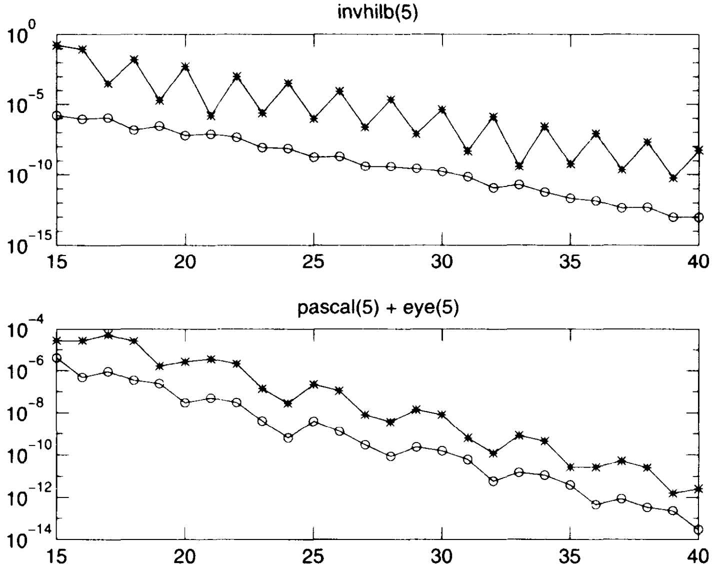


Figure 1.3. *Forward errors* $\|x-\widehat{x}\|_\infty/\|x\|_\infty$ ("$*$") *and relative residuals* $\|b-A\widehat{x}\|_\infty/(\|A\|_\infty\|\widehat{x}\|_\infty)$ ("o") *versus precision* $t = -\log_2 u$ *on the x axis.*

not monotonic. The reason for the pronounced oscillating behaviour of the relative error (but not the residual) for the inverse Hilbert matrix is not clear.

An example in which increasing the precision by several bits does not improve the accuracy is the evaluation of

$$y = x + a\sin(bx), \qquad x = \frac{1}{7}, \quad a = 10^{-8}, \quad b = 2^{24}. \tag{1.8}$$

Figure 1.4 plots t versus the absolute error, for precisions $u = 2^{-t}$, $t = 10{:}40$. Since $a\sin(bx) \approx -8.55\times10^{-9}$, for t less than about 20 the error is dominated by the error in representing $x = 1/7$. For $22 \le t \le 31$ the accuracy is (exactly) constant! The plateau over the range $22 \le t \le 31$ is caused by a fortuitous rounding error in the addition: in the binary representation of the exact answer the 23rd to 32nd digits are 1s, and in the range of t of interest the final rounding produces a number with a 1 in the 22nd bit and zeros beyond, yielding an unexpectedly small error that affects only bits 33rd onwards.

A more contrived example in which increasing the precision has no beneficial effect on the accuracy is the following evaluation of $z = f(x)$:

```
$y$ = abs(3($x$-0.5)-0.5)/25
if $y$ = 0
   $z$ = 1
```

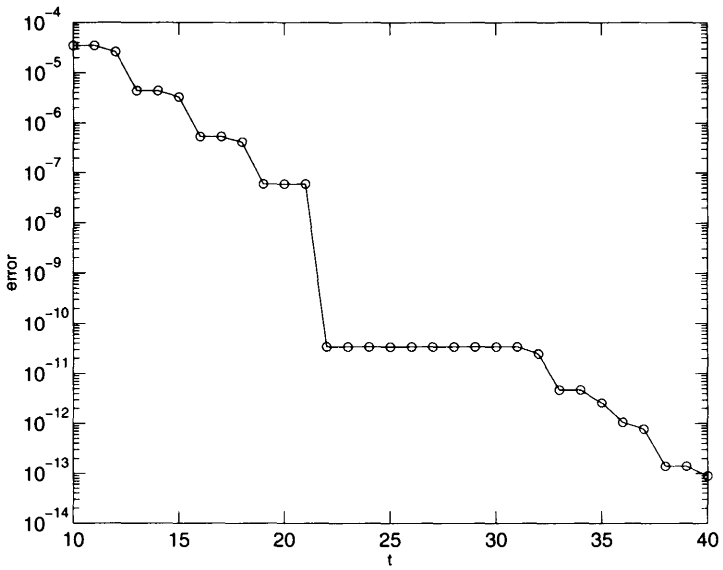


Figure 1.4. *Absolute error versus precision,* $t = -\log_2 u$.

```
else
    $z = e^y$    % Store to inhibit extended precision evaluation.
    $z = (z - 1)/y$
end
```

In exact arithmetic, $z = f(2/3) = 1$, but in Fortran 90 on a Sun SPARCstation and on a 486DX workstation, $\widehat{z} = fl(f(2/3)) = 0.0$ in both single and double precision arithmetic. A further example is provided by the "innocuous calculation" of §1.12.2, in which a step function is computed in place of $f(x) = x$ for a wide range of precisions.

It is worth stressing that *how* precision is increased can greatly affect the results obtained. Increasing the precision without preserving important properties such as monotonicity of rounding can vitiate an otherwise reliable algorithm. Increasing the precision without maintaining a correct relationship among the precisions in different parts of an algorithm can also be harmful to the accuracy.

1.14. Cancellation of Rounding Errors

It is not unusual for rounding errors to cancel in stable algorithms, with the result that the final computed answer is much more accurate than the intermediate quantities. This phenomenon is not universally appreciated, perhaps because we tend to look at the intermediate numbers in an algorithm only when something is wrong, not when the computed answer is satisfactory. We describe two examples. The first is a very short and rather unusual computation, while the second involves a well-known algorithm for computing a standard matrix decomposition.

1.14.1. Computing $(e^x - 1)/x$

Consider the function $f(x) = (e^x - 1)/x = \sum_{i=0}^{\infty} x^i/(i+1)!$, which arises in various applications. The obvious way to evaluate f is via the algorithm

```
% Algorithm 1.
if $x = 0$
  $f = 1$
else
  $f = (e^x - 1)/x$
end
```

This algorithm suffers severe cancellation for $|x| \ll 1$, causing it to produce an inaccurate answer (0 instead of 1, if x is small enough) . Here is an alternative:

```
% Algorithm 2.
$y = e^x$
if $y = 1$
  $f = 1$
else
  $f = (y - 1)/\log y$
end
```

At first sight this algorithm seems perverse, since it evaluates both exp and log instead of just exp. Some results computed in MATLAB are shown in Table 1.2. All the results for Algorithm 2 are correct in all the significant figures shown, except for $x = 10^{-15}$, when the last digit should be 1. On the other hand, Algorithm 1 returns answers that become less and less accurate as x decreases.

To gain insight we look at the numbers in a particular computation with $x = 9 \times 10^{-8}$ and $u = 2^{-24} \approx 6 \times 10^{-8}$, for which the correct answer is 1.00000005 to the significant digits shown. For Algorithm 1 we obtain a completely inaccurate result, as expected:

$$fl\left(\frac{e^x - 1}{x}\right) \equiv fl\left(\frac{1.19209290 \times 10^{-7}}{9.00000000 \times 10^{-8}}\right) = 1.32454766.$$

Table 1.2. *Computed values of* $(e^x - 1)/x$ *from Algorithms* 1 *and* 2.

x	Algorithm 1	Algorithm 2
10^{-5}	1.000005000006965	1.000005000016667
10^{-6}	1.000000499962184	1.000000500000167
10^{-7}	1.000000049433680	1.000000050000002
10^{-8}	$9.999999939225290 \times 10^{-1}$	1.000000005000000
10^{-9}	1.000000082740371	1.000000000500000
10^{-10}	1.000000082740371	1.000000000050000
10^{-11}	1.000000082740371	1.000000000005000
10^{-12}	1.000088900582341	1.000000000000500
10^{-13}	$9.992007221626408 \times 10^{-1}$	1.000000000000050
10^{-14}	$9.992007221626408 \times 10^{-1}$	1.000000000000005
10^{-15}	1.110223024625156	1.000000000000000
10^{-16}	0	

Algorithm 2 produces a result correct in all but the last digit:

$$fl\left(\frac{e^x - 1}{\log e^x}\right) \equiv fl\left(\frac{1.19209290 \times 10^{-7}}{1.19209282 \times 10^{-7}}\right) = 1.00000006.$$

Here are the quantities that would be obtained by Algorithm 2 in exact arithmetic (correct to the significant digits shown):

$$\frac{e^x - 1}{\log e^x} \equiv \frac{9.00000041 \times 10^{-8}}{9.00000001 \times 10^{-8}} = 1.00000005.$$

We see that Algorithm 2 obtains very inaccurate values of $e^x - 1$ and $\log e^x$, but the ratio of the two quantities it computes is very accurate. Conclusion: errors cancel in the division in Algorithm 2.

A short error analysis explains this striking cancellation of errors. We assume that the exp and log functions are both computed with a relative error not exceeding the unit roundoff u. The algorithm first computes $\widehat{y} = e^x(1+\delta)$, $|\delta| \le u$. If $\widehat{y} = 1$ then $e^x(1+\delta) = 1$, so

$$x = -\log(1+\delta) = -\delta+\delta^2/2-\delta^3/3+\dots, \qquad |\delta| \le u,$$

which implies that the correctly rounded value of $f(x) = 1+x/2+x^2/6+ \dots$ is 1, and so f has been evaluated correctly, to the working precision, If $\widehat{y} \ne 1$ then[7], using (1.1),

$$\widehat{f} = fl\big((\widehat{y} - 1)/\log \widehat{y}\big) = \frac{(\widehat{y} - 1)(1+\epsilon_1)}{\log \widehat{y}\,(1+\epsilon_2)}(1+\epsilon_3), \tag{1-9}$$

---

[7] The analysis from this point on assumes the use of a guard digit in subtraction (see §2.4); without a guard digit Algorithm 2 is not highly accurate.

where $|\epsilon_i| \le u$, i = 1:3. Defining $\upsilon = \widehat{y} - 1$, we have

$$\begin{aligned} g(\widehat{y}) &:= \frac{\widehat{y}-1}{\log \widehat{y}} = \frac{v}{\log(1+v)} \\ &= \frac{v}{v - v^2/2 + v^3/3 - \cdots} = \frac{1}{1 - v/2 + v^2/3 - \cdots} \\ &= 1 + \frac{v}{2} + O(v^2). \end{aligned}$$

For small x ($y \approx 1$)

$$g(\widehat{y}) - g(y) \approx \frac{\widehat{y}-y}{2} \approx \frac{e^x\delta}{2} \approx \frac{\delta}{2} \approx \frac{g(y)\delta}{2}.$$

From (1.9) it follows that $\widehat{f}$ approximates f with relative error at most about 3.5u.

The details of the analysis obscure the crucial property that ensures its success. For small x, neither $\widehat{y}$ - 1 nor $\log \widehat{y}$ agrees with its exact arithmetic counterpart to high accuracy. But $(\widehat{y} - 1)/\log \widehat{y}$ is an extremely good approximation to $(y-1)/\log y$ near y=1, because the function $g(y) = (y-1)/\log y$ varies so slowly there (g has a removable singularity at 1 and $g'(1) = 1$). In other words, the errors in $\widehat{y}$ - 1 and $\log \widehat{y}$ almost completely cancel.

1.14.2. QR Factorization

Any matrix $A \in \mathbb{R}^{m\times n}$, $m \ge n$, has a QR factorization $A = QR$, where $Q \in \mathbb{R}^{m\times}$ has orthonormal columns and $R \in \mathbb{R}^{n\times x}$ is upper trapezoidal ($r_{ij} = 0$ for $i > j$). One way of computing the QR factorization is to premultiply A by a sequence of Givens rotations-orthogonal matrices G that differ from the identity matrix only in a 2×2 principal submatrix, which has the form

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}.$$

With $A_1 := A$, a sequence of matrices A_k satisfying $A_k = G_kA_{k-1}$ is generated. Each A_k, has one more zero than the last, so $A_p = R$ for $p = n(m - (n + 1)/2)$. To be specific, we will assume that the zeros are introduced in the order (n,1). (n - 1,1), . . . , (2,1); (n,2), . . . , (3,2); and so on.

For a particular 10×6 matrix A, Figure 1.5 plots the relative errors $\|A_k - A_k\|_2/\|A\|_2$, where A_k, denotes the matrix computed in single precision arithmetic ($u \approx 6 \times 10^{-8}$). We see that many of the intermediate matrices are very inaccurate, but the final computed $\widehat{R}$ has an acceptably small relative error, of order u. Clearly, there is heavy cancellation of errors on the last few stages of the computation. This matrix $A \in \mathbb{R}^{10\times 6}$ was specially chosen,

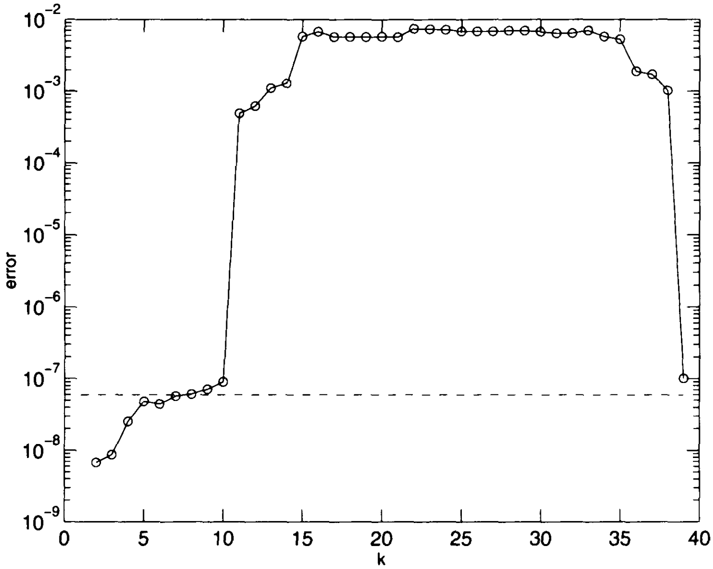


Figure 1.5. *Relative errors* $\|A_k-\widehat{A}_k\|_2/\|A\|_2$ *for Givens QR factorization. The dotted line is the unit roundoff level.*

following a suggestion of Wilkinson [1100, 1985], as a full matrix such that $\|A\|_2 \approx 1$ and A_{10} has the form

$$A_{10} = \begin{bmatrix} \widetilde{a}_{11} & \widetilde{a}_{12} & \widetilde{A}(1,3\colon n) \\ 0 & 1 & \widetilde{A}(2,3\colon n) \\ 0 & y & \widetilde{A}(3\colon m,3\colon n) \end{bmatrix},$$

Because y is at the roundoff level, the computed $\widehat{y}$ is the result of severe subtractive cancellation and so is dominated by rounding errors. Consequently, the computed Givens rotations $\widetilde{G}_{10}, \dots, \widetilde{G}_{17}$, whose purpose is to zero the vector $\widehat{y}$, and which are determined by ratios involving the elements of $\widehat{y}$, bear little relation to their exact counterparts, causing $\widehat{A}_k$ to differ greatly from A_k for $k = 11, 12, \dots$.

To shed further light on this behaviour, we note that the Givens QR factorization is perfectly backward stable; that is, the computed R is the exact R factor of $A + \Delta A$, where $\|\Delta A\|_2 < cu\|A\|_2$, with c a modest constant depending on the dimensions (Theorem 18.9). By invoking a perturbation result for the QR factorization (namely (18.27)) we conclude that $\|R-\widehat{R}\|_2/\|A\|_2$ is bounded by a multiple of $\kappa_2(A)u$. Our example is constructed so that $\kappa_2(A)$ is small (≈ 24), so we know a priori that the graph in Figure 1.5 must eventually dip down to the unit roundoff level.

We also note that $\|Q - \widehat{Q}\|_2$ is of order u in this example, as again we can show it must be from perturbation theory. Since Q is a product of Givens rotations, this means that even though some of the intermediate Givens rotations are very inaccurate, their product is highly accurate, so in the formation of Q, too, there is extensive cancellation of rounding errors.

1.15. Rounding Errors Can Be Beneficial

An old method for computing the largest eigenvalue (in absolute value) of a matrix A and the corresponding eigenvector is the power method, which consists of repeatedly multiplying a given starting vector by A. With scaling to avoid underflow and overflow, the process in its simplest form is

```
% Choose a starting vector $x$.
while not converged
    $x := Ax$
    $x := x/\|x\|_\infty$
end
```

The theory says that if A has a unique eigenvalue of largest modulus and x is not deficient in the direction of the corresponding eigenvector υ, then the power method converges to a multiple of υ (at a linear rate).

Consider the matrix

$$A = \begin{bmatrix} 0.4 & -0.6 & 0.2 \\ -0.3 & 0.7 & -0.4 \\ -0.1 & -0.4 & 0.5 \end{bmatrix},$$

which has eigenvalues 0, 0.4394 and 1.161 (correct to the digits shown) and an eigenvector $[1, 1, 1]^T$ corresponding to the eigenvalue zero. If we take $[1,1,1]^T$ as the starting vector for the power method then, in principle, the zero vector is produced in one step, and we obtain no indication of the desired dominant eigenvalue-eigenvector pair. However, when we carry out the computation in MATLAB, the first step produces a vector with elements of order 10^{-16} and we obtain after 38 iterations a good approximation to the dominant eigenpair. The explanation is that the matrix A cannot be stored exactly in binary floating point arithmetic. The computer actually works with $A + \Delta A$ for a tiny perturbation ΔA, and the dominant eigenvalue and eigenvector of $A + \Delta A$ are very good approximations to those of A. The starting vector $[1,1,1]^T$ contains a nonzero (though tiny) component of the dominant eigenvector of $A + \Delta A$. This component grows rapidly under multiplication by $A + \Delta A$, helped by rounding errors in the multiplication, until convergence to the dominant eigenvector is obtained.

Perhaps an even more striking example of beneficial effects of rounding errors is in inverse iteration, which is just the power method applied to the shifted and inverted matrix $(A-\mu I)^{-1}$. The shift μ is usually an approximate eigenvalue. The closer μ is to an eigenvalue, the more nearly singular $A - \mu I$ is, and hence the larger the error in computing $y = (A - \mu I)^{-1} x$ (which is done by solving $(A-\mu I)y = x$). However, it can be shown that the error lies almost entirely in the direction of the required eigenvector, and so is harmless; see, for example, Parlett [820, 1980, §4.31] or Golub and Van Loan [470, 1989, §7.6.1].

1.16. Stability of an Algorithm Depends on the Problem

An algorithm can be stable as a means for solving one problem but unstable when applied to another problem. One example is the modified Gram-Schmidt method, which is stable when used to solve the least squares problem but can give poor results when used to compute an orthonormal basis of a matrix (see §§518.7 and 19.3).

A lesser known and much simpler example is Gaussian elimination (GE) without pivoting for computing the determinant of an upper Hessenberg matrix. A square matrix A is upper Hessenberg if $a_{ij} = 0$ for $i > j + 1$. GE transforms A to upper triangular form by $n - 1$ row eliminations. one for each of the boxed entries in this 4 × 4 illustration:

$$A = \begin{bmatrix} \times & \times & \times & \times \\ \boxed{\times} & \times & \times & \times \\ 0 & \boxed{\times} & \times & \times \\ 0 & 0 & \boxed{\times} & \times \end{bmatrix} \longrightarrow \begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \end{bmatrix} = U.$$

The determinant of A is given by the product of the diagonal elements of U. It is easy to show that this is a stable way to evaluate $\det(A)$, even though arbitrarily large multipliers may arise during the elimination. Note. first, that, if $A^{(k)}$ denotes the matrix at the start of the kth stage ($A^{(1)} = A$), then

$$u_{kk} = a_{kk}^{(k)} = a_{kk}^{(k-1)} - \frac{a_{k,k-1}^{(k-1)} a_{k-1,k}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}} = a_{kk} - \frac{a_{k,k-1} a_{k-1,k}^{(k-1)}}{a_{k-1,k-1}^{(k-1)}},$$

because the kth row of $A^{(k-1)}$ is the same as the kth row of A. In floating point arithmetic the model (1.1) shows that the computed $\widehat{a}_{ij}^{(k)}$ satisfy

$$\begin{aligned} \widehat{u}_{kk} = \widehat{a}_{kk}^{(k)} &= \left(a_{kk} - \frac{a_{k,k-1} \widehat{a}_{k-1,k}^{(k-1)}}{\widehat{a}_{k-1,k-1}^{(k-1)}} (1+\epsilon_1^{(k)})(1+\epsilon_2^{(k)}) \right) (1+\epsilon_3^{(k)}) \\ &= a_{kk}(1+\epsilon_3^{(k)}) - \frac{\left[a_{k,k-1}(1+\epsilon_1^{(k)})(1+\epsilon_2^{(k)})(1+\epsilon_3^{(k)})\right] \widehat{a}_{k-1,k}^{(k-1)}}{\widehat{a}_{k-1,k-1}^{(k-1)}}, \end{aligned}$$

Table 1.3. *Results from GE without pivoting on an upper Hessenberg matrix.*

	Exact	Computed	Relative error
x:	$\begin{bmatrix} 1.0000 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}$	$\begin{bmatrix} 2.3842 \\ 1.0000 \\ 1.0000 \\ 1.0000 \end{bmatrix}$	1.3842
$\det(A)$:	2.0000	2.0000	1.9209×10^{-8}

where $|\epsilon_i^{(k)}| \le u$, $i = 1{:}3$. This equation says that the computed diagonal elements $\widehat{u}_{kk}$ are the exact diagonal elements corresponding not to A, but to a matrix obtained from A by changing the diagonal elements to $a_{kk}(1 + \epsilon_3^{(k)})$ and the subdiagonal elements to $a_{k,k-1}(1+\epsilon_1^{(k)})(1+\epsilon_2^{(k)})(1+\epsilon_3^{(k)})$. In other words, the computed $\widehat{u}_{kk}$ are exact for a matrix differing negligibly from A. The computed determinant d, which is given by

$$\widehat{d} = fl(\widehat{u}_{11} \dots \widehat{u}_{nn}) = \widehat{u}_{11} \dots \widehat{u}_{nn}(1+\eta_1)\dots(1+\eta_n), \qquad |\eta_i| \le u,$$

is therefore a tiny relative perturbation of the determinant of a matrix differing negligibly from A. so this method for evaluating $\det(A)$ is numerically stable (in the mixed forward backward error sense of (1.2)).

However, if we use GE without pivoting to solve an upper Hessenberg linear system then large multipliers can cause the solution process to be unstable. If we try to extend the analysis above we find that the computed LU factors (as opposed to just the diagonal of U) do not, as a whole, necessarily correspond to a small perturbation of A.

A numerical example illustrates these ideas. Let

$$A = \begin{bmatrix} \alpha & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 0 & 1 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

We took $a = 10^{-7}$ and $b = Ae(e = [1,1,1,1]^T)$ and used GE without pivoting in single precision arithmetic ($u \approx 6 \times 10^{-8}$) to solve $Ax = b$ and compute $\det(A)$. The computed and exact answers are shown to five significant figures in Table 1.3. Not surprisingly, the computed determinant is very accurate. But the computed solution to $Ax = b$ has no correct figures in its first component. This reflects instability of the algorithm rather than ill conditioning of the problem because the condition number $\kappa_\infty(A) = 16$. The source of the instability is the large first multiplier, $a_{21}/a_{11} = 10^7$.

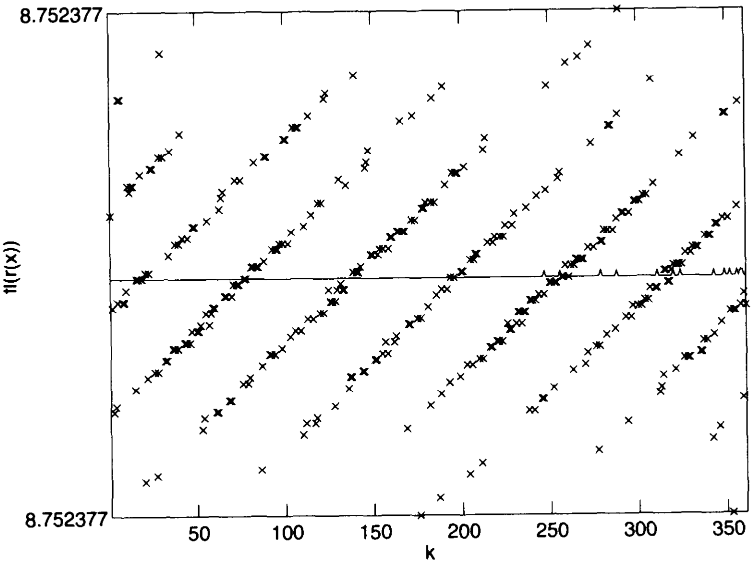


Figure 1.6. *Values of rational function r(x) computed by Horner's rule (marked as "×"), for* $x = 1.606 + (k - 1)2^{-52}$; *solid line is the "exact" r(x).*

1.17. Rounding Errors Are Not Random

Rounding errors, and their accumulated effect on a computation, are not random. This fact underlies the success of many computations, including some of those described earlier in this chapter. The validity of statistical analysis of rounding errors is discussed in §2.6. Here we simply give a revealing numerical example (due to W. Kahan).

Define the rational function

$$r(x) = \frac{622 - x(751 - x(324 - x(59 - 4x)))}{112 - x(151 - x(72 - x(14 - x)))},$$

which is expressed in a form corresponding to evaluation of the quartic polynomials in the numerator and denominator by Horner's rule. We evaluated $r(x)$ by Horner's rule in double precision arithmetic for 361 consecutive floating point numbers starting with $a = 1.606$, namely $x = a + (k - 1)2^{-52}$, $k = 1{:}361$; the function $r(x)$ is virtually constant on this interval. Figure 1.6 plots the computed function values together with a much more accurate approximation to $r(x)$ (computed from a continued fraction representation). The striking pattern formed by the values computed by Horner's rule shows clearly that the rounding errors in this example are not random.

1.18. Designing Stable Algorithms

There is no simple recipe for designing numerically stable algorithms. While this helps to keep numerical analysts in business (even in proving each other's algorithms to be unstable!) it is not good news for computational scientists in general. The best advice is to be *aware* of the need for numerical stability when designing an algorithm and not to concentrate solely on other issues, such as computational cost and parallelizability.

A few guidelines can be given.

1. Try to avoid subtracting quantities contaminated by error (though such subtractions may be unavoidable).

2. Minimize the size of intermediate quantities relative to the final solution. The reason is that if intermediate quantities are very large then the final answer may be the result of damaging subtractive cancellation. Looked at another way, large intermediate numbers swamp the initial data, resulting in loss of information. The classic example of an algorithm where this consideration is important is Gaussian elimination (§9.2), but an even simpler one is recursive summation (§4.2).

3. Look for different formulations of a computation that are mathematically but not numerically equivalent. For example, the classical Gram-Schmidt method is unstable, but a trivial modification produces the stable modified Gram-Schmidt (MGS) method (§18.7). There are two ways of using the MGS method to solve a least squares problem, the more obvious of which is unstable (§19.3).

4. It is advantageous to express update formulae as

   new-value = old-value + small-correction

   if the small correction can be computed with many correct significant figures. Numerical methods are often naturally expressed in this form; examples include methods for solving ordinary differential equations, where the correction is proportional to a step size, and Newton's method for solving a nonlinear system. A classic example of the use of this update strategy is in iterative refinement for improving the computed solution to a linear system $Ax = b$, in which by computing residuals $r = b - Ay$ in extended precision and solving update equations that have the residuals as right-hand sides a highly accurate solution can be computed: see Chapter 11. For another example (in which the correction is not necessarily small), see Problem 2.8.

5. Use only well-conditioned transformations of the problem. In matrix computations this amounts to multiplying by orthogonal matrices where

possible, instead of nonorthogonal, and possibly, ill-conditioned matrices. See §6.2 for a simple explanation of this advice in terms of norms.

6. Take precautions to avoid unnecessary overflow and underflow (see §25.8).

Concerning the second point, good advice is to look at the numbers generated during a computation. This was common practice in the early days of electronic computing. On some machines it was unavoidable because the contents of the store were displayed on lights or monitor tubes! Wilkinson gained much insight into numerical stability by inspecting the progress of an algorithm, and sometimes altering its course (for an iterative process with parameters): "Speaking for myself I gained a great deal of experience from user participation, and it was this that led to my own conversion to backward error analysis" [1099, 1980, pp. 112-113] see also [1083, 1955]). It is ironic that with the wealth of facilities we now have for tracking the progress of numerical algorithms (multiple windows in colour, graphical tools, fast printers) we often glean less than Wilkinson and his co-workers did from mere paper tape and lights.

1.19. Misconceptions

Several common misconceptions and myths have been dispelled in this chapter (none of them for the first time-see the Notes and References). We highlight them in the following list.

1. Cancellation in the subtraction of two nearly equal numbers is always a bad thing (§1.7).

2. Rounding errors can overwhelm a computation only if vast numbers of them accumulate (§1.11).

3. A short computation free from cancellation, underflow, and overflow must be accurate (§1.12).

4. Increasing the precision at which a computation is performed increases the accuracy of the answer (§1.13).

5. The final computed answer from an algorithm cannot be more accurate than any of the intermediate quantities, that is, errors cannot cancel (§1.14).

6. Rounding errors can only hinder, not help, the success of a computation (§1.15).

1.20. Rounding Errors in Numerical Analysis

Inevitably, much of this book is concerned with numerical linear algebra, because this is the area of numerical analysis in which the effects of rounding errors are most important and have been most studied. In nonlinear problems rounding errors are often not a major concern because other forms of error dominate. Although we give examples for numerical methods involving derivatives and integrals (for example, Euler's met hod in §4.3 and quadrature in Problem 3.12), it is beyond our scope to give a treatment of the effects and influence of rounding errors on these methods. We do. however, give some selected references to the literature. grouped by subject area.

Approximation theory: Clenshaw [212, 19551], Cox [249, 19721], [250, 19751], [251, 1978], Cox and Harris [253, 1989], de Boor [272, 19721], and de Boor and Pinkus [273, 1977].

Chaos and dynamical systems: Cipra [211, 1988], Coomes, Koçak, and Palmer [242, 1995], Corless [246. 1992], [247, 1992], Hammel, Yorke, and Grebogi [499, 1988], and Sanz-Serna and Larsson [893, 1993].

Nonlinear equations: Dennis and Walker [302, 1984], Spellucci [933, 1980], and Wozniakowski [1111, 1977].

Optimization: Dennis and Schnabel [300, 1983], Fletcher [377, 1986], [379, 1988], [380, 1993], [381, 1994]. Gill, Murray, and Wright [447, 1981], Gurwitz [490, 1992], Müller-Merbach [783, 1970], and Wolfe [1108, 1965].

Ordinary differential equation initial value problems: Henrici [517, 1962], [518, 1963], [519, 1964], D. J. Higham [525, 1991]. Sanz-Serna [891, 1992, §12], Sanz-Serna and Calvo [892, 1994], and Shampine [910, 1994, §3.3, §5.6].

Partial differential equations: Ames [14, 1977], Birkhoff and Lynch [101, 1984], Canuto, Hussaini, Quarteroni, and Zang [183, 1988], Douglas [319, 1959], Forsythe and Wasow [397, 1960], Richtmyer and Morton [872, 1967, §1.8], and Trefethen and Trummer [1020, 1987].

Quadrature: Davis and Rabinowitz [267, 1984, §4.2], Lyness [717, 1969], and Piessens et al. [832, 1983, §3.4.3.1].

1.21. Notes and References

The term "correct significant digits" is rarely defined in textbooks: it is apparently assumed that the definition is obvious. One of the earliest books on numerical analysis, by Scarborough [897, 1950] (first edition 1930), is noteworthy for containing theorems describing the relationship between correct significant digits and relative error.

The first definition of correct significant digits in §1.2 is suggested by Hildebrand [571, 1974, §1.41], who notes its weaknesses.

For a formal proof and further explanation of the fact that precision does not limit accuracy see Priest [844, 1992].

It is possible to develop formal definitions of numerical stability, either with respect to a particular problem, as is frequently done in research papers, or for a very general class of problems, as is done, for example, by de Jong [274, 1977]. Except in §7.6, we do not give formal definitions of stability in this book, preferring instead to adapt informally the basic notions of backward and forward stability to each problem, and thereby to minimize the amount of notation and abstraction.

Backward error analysis was systematically developed, exploited, and popularized by Wilkinson in the 1950s and 1960s in his research papers and, in particular, through his books [1088, 1963], [1089, 1965] (for more about the books see the Notes and References for Chapter 2). Backward error ideas had earlier appeared implicitly in papers by von Neumann and Goldstine (1057, 1947] and Turing [1027, 1948], both of which deal with the solution of linear systems, and explicitly in an unpublished technical report of Givens [451, 1954] on the solution of the symmetric eigenproblem by reduction to tridiagonal form followed by the use of Sturm sequences. The concept of backward error is not limited to numerical linear algebra. It is used, for example, in the numerical solution of differential equations; see Eirola [349, 1993], Enright [352, 1989], and Shampine [910, 1994, §2.2], in addition to the references of Sanz-Serna and his co-authors cited in §1.20. Backward error is also used in understanding chaotic behaviour of iterations; see the references in §1.20.

Conditioning of problems has been studied by numerical analysts since the 1940s but the first general theory was developed by Rice [871, 1966]. In numerical linear algebra, developing condition numbers is part of the subject of perturbation theory, on which there is a large literature.

The solution of a quadratic equation is a classic problem in numerical analysis. In 1969 Forsythe [393, 1969] pointed out “the near absence of algorithms to solve even a quadratic equation in a satisfactory way on actually used digital computer systems” and he presented specifications suggested by Kahan for a satisfactory solver. Similar, but less technical, presentations are given by Forsythe [392, 1969] and Forsythe, Malcolm, and Moler [395, 1977, §2.6]. Kahan [627, 1972] and Sterbenz [938, 1974] both present algorithms for solving a quadratic equation, accompanied by error analysis.

For more details of algorithms for computing the sample variance and their error analysis, see Chan and Lewis [195, 1979], Chan, Golub, and LeVeque [194, 1983], Barlow [61, 1991], and the references therein. Good general references on computational aspects of statistics are Kennedy and Gentle [649, 1980] and Thisted [1000. 1988].

The issues of conditioning and numerical stability play a role in any discipline in which finite precision computation is performed, but the understanding of these issues is less well developed in some disciplines than in others. In geometric computation, for example, there has been much interest since the late 1980s in the accuracy and robustness of geometric algorithms; see Milenkovic [754, 1988], Hoffmann [576, 1989], and Priest [843, 1991], [844, 1992].

It was after discovering Lemma 1.1 that Wilkinson began to develop backward error analysis systematically in the 1950s. He explains that in solving eigenproblems $Ax = \lambda x$ by deflation, the residual of the computed solution, $r := A\overline{x} - \overline{\lambda}\overline{x}$ (with the normalization $\overline{x}^T\overline{x} = 1$) was "always at noise level relative to A" [1101, 1986]. He continues, "After some years' experience of this I happened. almost by accident, to observe that . . . $(A - r\overline{x}^T)\overline{x} = \overline{\lambda}\overline{x}$ . . . In other words $\overline{\lambda}$ and $\overline{x}$ were exact for a matrix $A - r\overline{x}^T$ and since $\|r\overline{x}^T\|_2 = \|r\|_2$, this meant that they were exact for a matrix differing from A at the noise level of the computer." For further details see [1101, 1986] or [1100, 1985].

The numerical stability of Cramer's rule for 2 × 2 systems has been investigated by Moler [768, 1974] and Stummel [964, 1981, §3.3].

The example in §1.12.2 is taken from the *HP-15C Advanced Functions Handbook* [523, 1982], and a similar example is given by Kahan [629, 1980]. For another approach to analysing this "innocuous calculation" see Problem 3.11. The "$f(2/3)$" example in §1.13 is also taken from [629, 1980], in which Kahan states three "anti-theorems" that are included among our misconceptions in §1.19.

The example (1.8) is adapted from an example of Sterbenz [938, 1974, p. 220], who devotes a section to discussing the effects of rerunning a computation at higher precision.

The function expm1 := $e^x - 1$ is provided in some floating point processors and mathematics libraries as a more accurate alternative to forming e^x and subtracting 1 [991, 1992]. It is important in the computation of sinh and tanh, for example (since $\sinh x = e^{-x}(e^{2x} - 1)/2$). Algorithm 2 in §1.14.1 is due to Kahan [629, 1980].

The instability. and stability of GE without pivoting applied to an upper Hessenberg matrix (§1.16) was first pointed out and explained by Wilkinson [1084, 1960]; Parlett [818, 1965] also gives a lucid discussion. In the 1950s and 1960s. prior to the development of the QR algorithm, various methods were proposed for the nonsymmetric eigenvalue problem that involved transforming a matrix to Hessenberg form H and then finding the zeros of the characteristic polynomial $\det(H - \lambda I)$. The most successful method of this type was Laguerre's iteration. described by Parlett [817, 1964], and used in conjunction with Hyman's method for evaluating $\det(H - \lambda I)$. Hyman's method is described in §13.5.1.

Classic papers dispensing good advice on the dangers inherent in numerical computation are the "pitfalls" papers by Stegun and Abramowitz [937, 1956] and Forsythe [394, 1970]. The book *Numerical Methods That Work* by Acton [4, 1970] must also be mentioned as a fount of hard-earned practical advice on numerical computation (look carefully and you will see that the front cover includes a faint image of the word "Usually" before "Work"). If it is not obvious to you that the equation $x^2 - 10x + 1 = 0$ is best thought of as a nearly linear equation for the smaller root, you will benefit from reading Acton (see p. 58). Everyone should read Acton's "Interlude: What *Not* To Compute" (pp. 245-257).

Finally, we mention the paper "How to Get Meaningless Answers in Scientific Computation (and What to Do About it)" by Fox [401, 1971]. Fox, a contemporary of Wilkinson, founded the Oxford Computing Laboratory and was for many years Professor of Numerical Analysis at Oxford. In this paper he gives numerous examples in which incorrect answers are obtained from plausible numerical methods (many of the examples involve truncation errors as well as rounding errors). The section titles provide a list of reasons why you might compute worthless answers:

- Your problem might be ill conditioned.
- Your method might be unstable.
- You expect too much "analysis" from the computers.
- Your intuition fails you.
- You accept consistency too easily.
- A successful method may fail in slightly different circumstances.
- Your test examples may be too special.

Fox estimates [401, 1971, p. 296] that "about 80 per cent of all the results printed from the computer are in error to a much greater extent than the user would believe."

---

[8] This reason refers to using an inappropriate convergence test in an iterative process.

Problems

> *The road to wisdom?*
> *Well, it's plain and simple to express:*
> *Err*
> *and err*
> *and err again*
> *but less*
> *and less*
> *and less.*
>
> -PIET HEIN, *Grooks (1966)*

1.1. In error analysis it is sometimes convenient to bound $\widetilde{E}_{\mathrm{rel}}(\widehat{x}) = |x-\widehat{x}|/|\widehat{x}|$ instead of $E_{\mathrm{rel}}(\widehat{x}) = |x-\widehat{x}|/|x|$. Obtain inequalities between $E_{\mathrm{rel}}(\widehat{x})$ and $\widetilde{E}_{\mathrm{rel}}(\widehat{x})$.

1.2. (Skeel and Keiper [923, 1993, §1.2]) The number $y = e^{\pi\sqrt{163}}$ was evaluated at t digit precision for several values of t, yielding the values shown in the following table. which are in error in at most one unit in the least significant digit (the first two values are padded with trailing zeros):

t	y
10	262537412600000000
15	262537412640769000
20	262537412640768744.00
25	262537412640768744.0000000
30	262537412640768743.999999999999

Does it follow that the last digit before the decimal point is 4?

1.3. Show how to rewrite the following expressions to avoid cancellation for the indicated arguments.

1. $\sqrt{x+1}-1$, $x \approx 0$.
2. $\sin x - \sin y$, $x \approx y$.
3. $x^2 - y^2$, $x \approx y$.
4. $(1-\cos x)/\sin x$, $x \approx 0$.
5. $c = (a^2 + b^2 - 2ab\cos\theta)^{1/2}$, $a \approx b$, $|\theta| \ll 1$.

1.4. Give stable formulae for computing the square root $x + iy$ of a complex number $a + ib$.

1.5. [523, 1982] By writing $(1 + 1/n)^n = \exp(n \log(1 + 1/n))$, show how to compute $(1 + 1/n)^n$ accurately for large n. Assume that the log function is computed with a relative error not exceeding u. (Hint: adapt the technique used in §1.14.1.)

1.6. (Smith [928, 1975]) Type the following numbers into your pocket calculator, and look at them upside down (you or the calculator):

07734	The famous “_world” program
38079	Object
318808	Name
35007	Adjective
$57738.57734 \times 10^{40}$	Exclamation on finding a bug
3331	A high quality floating point arithmetic
$\sqrt{31,438,449}$	Fallen tree trunks

1.7. A condition number for the sample variance (1.4), here denoted by $V(x) : \mathbb{R}^n \to \mathbb{R}$, can be defined by

$$\kappa_C := \lim_{\epsilon \to 0} \sup \left\{ \frac{|V(x) - V(x + \Delta x)|}{\epsilon V(x)} : |\Delta x_i| \le \epsilon |x_i|,\ i = 1\colon n \right\}.$$

Show that

$$\kappa_C = 2 \frac{\sum_{i=1}^n |x_i - \overline{x}||x_i|}{(n-1)V(x)}.$$

This condition number measures perturbations in x componentwise. A corresponding normwise condition number is

$$\kappa_N := \lim_{\epsilon \to 0} \sup \left\{ \frac{|V(x) - V(x + \Delta x)|}{\epsilon V(x)} : \|\Delta x\|_2 \le \epsilon \|x\|_2 \right\}.$$

Show that

$$\kappa_N = 2 \frac{\|x\|_2}{\sqrt{(n-1)V(x)}} = 2 \left(1 + \frac{n}{n-1} \frac{\overline{x}^2}{V(x)} \right)^{1/2} \ge \kappa_C.$$

1.8. (Kahan, Muller, [781, 1989], Francois and Muller [406, 1991]) Consider the recurrence

$$x_{k+1} = 111 - (1130 - 3000/x_{k-1})/x_k, \qquad x_0 = 11/2, \quad x_1 = 61/11.$$

In exact arithmetic the x_k form a monotonically increasing sequence that converges to 6. Implement the recurrence on your computer or pocket calculator and compare the computed x_{34} with the true value 5.998 (to four correct significant figures). Explain what you see.

The following questions require knowledge of material from later chapters.

1.9. Cramer’s rule solves a 2×2 system $Ax = b$ according to

$$\begin{aligned} d &= a_{11}a_{22} - a_{21}a_{12}, \\ x_1 &= (b_1 a_{22} - b_2 a_{12})/d, \\ x_2 &= (a_{11}b_2 - a_{21} b_1)/d. \end{aligned}$$

Show that, assuming *d* is computed exactly (this assumption has little effect on the final bounds), the computed solution $\widehat{x}$ satisfies

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \le \gamma_3 \operatorname{cond}(A,x), \qquad \|b - A\widehat{x}\|_\infty \le \gamma_3 \operatorname{cond}(A^{-1})\|b\|_\infty,$$

where $\gamma_3 = 3u/(1-3u)$, $\operatorname{cond}(A,x) = \|\,|A^{-1}||A||x|\,\|_\infty/\|x\|_\infty$ and $\operatorname{cond}(A) = \|\,|A^{-1}||A|\,\|_\infty$. This forward error bound is as small as that for a backward stable method (see §7.2, §7.6) so Cramer's rule is forward stable for 2 × 2 systems.

1.10. Show that the computed sample variance $\widehat{V} = fl(V(x))$ produced by the two-pass formula (1.4) satisfies

$$\frac{|V-\widehat{V}|}{V} \le (n+3)u + O(u^2).$$

(Note that this error bound does not involve the condition numbers κ_C or κ_N from Problem 1.7, at least in the first-order term. This is a rare instance of an algorithm that determines the answer more accurately than the data warrants!)

Chapter 2

Floating Point Arithmetic

From 1946-1948 *a great* deal *of quite detailed coding was done.*
The subroutines for floating-point arithmetic were . . .
produced by Alway and myself in 1947 . . .
They were almost certainly the earliest floating-point subroutines.

-J. H. WILKINSON, *Turing's Work at the*
National Physical Laboratory . . . (1980)

When discussing the floating-point capabilities of a new machine,
we always ask the manufacturer two questions:
Does the machine use IEEE arithmetic?
Does it support graceful underflow and provide
user control of rounding mode and exception flags?
Frequently the designer believes his machine is using IEEE arithmetic
when it is using only the IEEE formats without the other required features.

-*W.* J. CODY, *Floating-Point Standards-Theory and Practice* (1988)

Arithmetic on Cray computers is interesting because it is driven by a
motivation for the highest possible floating-point performance . . .
Addition on Cray computers does not have a *guard digit,*
and multiplication is even less accurate than addition . . .
At least Cray computers serve to keep numerical analysts on their toes!

-DAVID GOLDBERG[9], *Computer Arithmetic (1990)*

It is rather conventional to obtain a "realistic" estimate
of the possible overall error due to k *roundoffs,*
when k *is fairly large,*
by replacing k *by* $\sqrt{k}$ *in an expression for (or an estimate of)*
the maximum *resultant error.*

-F. B. HILDEBRAND, *introduction to Numerical Analysis* (1974)

---

[9] In Hennessy and Patterson [515, 1990, App. A].

2.1. Floating Point Number System

A floating point number system $F \subset \mathbb{R}$ is a subset of the real numbers whose elements have the form

$$y = \pm m \times \beta^{e-t}. \tag{2.1}$$

The system F is characterized by four integer parameters:

- the base β (sometimes called the radix),
- the *precision* t, and
- the *exponent range* $e_{\min} \le e \le e_{\max}$.

The *mantissa* m is an integer satisfying $0 \le m \le \beta^t - 1$. To ensure a unique representation for each $y \in F$ it is assumed that $m \ge \beta^{t-1}$ if $y \neq 0$, so that the system is *normalized*. The range of the nonzero floating point numbers in F is given by $\beta^{e_{\min}-1} \le |y| \le \beta^{e_{\max}}(1 - \beta^{-t})$. Values of the parameters for some machines of interest are given in Table 2.1 (the unit roundoff u is defined on page 42).

Note that an alternative (and more common) way of expressing y is

$$y = \pm\beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t}\right) = \pm\beta^e \times .d_1 d_2 \ldots d_t, \tag{2.2}$$

where each digit d_i satisfies $0 \le d_i \le \beta - 1$, and $d_1 \neq 0$ for normalized numbers. We prefer the more concise representation (2.1), which we usually find easier to work with. This "nonpositional" representation has pedagogical advantages. being entirely integer based and therefore simpler to grasp. In the representation (2.2). d_1 is called the most *significant digit* and d_t the *least significant digit.*

It is important to realize that the floating point numbers are not equally spaced. If $\beta = 2$, $t = 3$, $e_{\min} = -1$, and $e_{\max} = 3$ then the nonnegative floating point numbers are

0, 0.25, 0.3125, 0.3750, 0.4375, 0.5, 0.625, 0.750, 0.875,
1.0, 1.25, 1.50, 1.75. 2.0, 2.5, 3.0. 3.5, 4.0, 5.0, 6.0, 7.0.

They can be represented pictorially as follows:

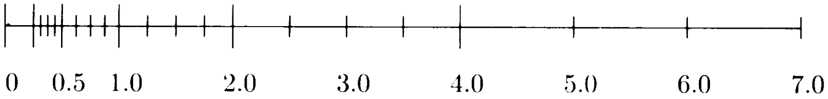


Table 2.1. *Floating point arithmetic parameters.*

Machine and arithmetic	β	t	$e$min	$e$max	u
Cray-1 single	2	48	-8192	8191	4×10^{-15}
Cray-1 double	2	96	-8192	8191	1×10^{-29}
DEC VAX G format, double	2	53	-1023	1023	1×10^{-16}
DEC VAX D format, double	2	56	-127	127	1×10^{-17}
HP 28 and 48G calculators	10	12	-499	499	5×10^{-12}
IBM 3090 single	16	6	-64	63	5×10^{-7}
IBM 3090 double	16	14	-64	63	1×10^{-16}
IBM 3090 extended	16	28	-64	63	2×10^{-33}
IEEE single	2	24	-125	128	6×10^{-8}
IEEE double	2	53	-1021	1024	1×10^{-16}
IEEE extended (typical)	2	64	-16381	16384	5×10^{-20}

Notice that the spacing of the floating point numbers jumps by a factor 2 at each power of 2. The spacing can be characterized in terms of *machine epsilon*, which is the ϵ_M from 1.0 to the next larger floating point number. Clearly, $\epsilon_M = \beta^{1-t}$, and this is the spacing of the floating point numbers between 1.0 and β; the spacing of the numbers between 1.0 and $1/\beta$ is $\beta^{-t} = \epsilon_M/\beta$. The spacing at an arbitrary $x \in F$ is estimated by the following lemma.

Lemma 2.1. *The spacing between a normalized floating point number x and an adjacent normalized floating point number is at least $\beta^{-1}\epsilon_M|x|$ and at most $\epsilon_M|x|$ (unless x or the neighbour is zero).*

Proof. See Problem 2.2.

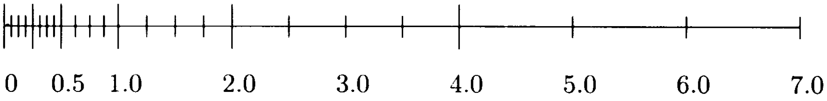
The system F can be extended by including *subnormal numbers* (also known as *denormalized numbers),* which, in the notation of (2.1), are the numbers

$$y = \pm m \times \beta^{e\min - t}, \qquad 0 < m < \beta^{t-1},$$

which have the minimum exponent and are not normalized (equivalently, in (2.2) $e = e\min$ and the most significant digit $d_1 = 0$). The subnormal numbers have fewer digits of precision than the normalized numbers. The smallest positive normalized floating point number is $\lambda = \beta^{e\min-1}$ while the smallest positive subnormal number is $\mu = \beta^{e\min-t} = \lambda\epsilon_M$. The subnormal numbers fill the gap between λ and 0 and are equally spaced, with spacing the same as that of the numbers of F between λ and $\beta\lambda$, namely $\lambda\epsilon_M = \beta^{e\min-t}$. For example, in the system illustrated above with $\beta = 2$, $t = 3$, $e\min = -1$, and $e\max = 3$, we have $\lambda = 2^{-2}$ and $\mu = 2^{-4}$, the subnormal numbers are

0.0625, 0.125, 0.1875,

and the complete number system can be represented as



Let $G \subset$ IR denote all real numbers of the form (2.1) with no restriction on the exponent e. If $x \in$ IR then $fl(x)$ denotes an element of G nearest to x, and the transformation $x \rightarrow fl(x)$ is called *rounding*. There are several ways to break ties when x is equidistant from two floating point numbers, including taking $fl(x)$ to be the number of larger magnitude (round away from zero) or the one with an even last digit d_t (round to even): the latter rule enjoys impeccable statistics [144, 1973]. For more on tie-breaking strategies see the Notes and References.

Although we have defined fl as a mapping onto G, we are only interested in the cases where it produces a result in F. We say that $fl(x)$ *overflows* if $|fl(x)| > \max\{|y| \ : \ y \in F\}$ and *underflows* if $0 < |fl(x)| < \min\{|y| \ : \ 0 \neq y \in F\}$.

The following result shows that every real number x lying in the range of F can be approximated by an element of F with a relative error no larger than $u = \frac{1}{2}\beta^{1-t}$. The quantity u is called the *unit roundoff*. It is the most useful quantity associated with F and is ubiquitous in the world of rounding error analysis.

Theorem 2.2. *If $x \in$ **IR** lies in the range of F then*

$$fl(x) = x(1 + \delta), \qquad |\delta| < u. \tag{2.3}$$

Proof. We can assume that $x > 0$. Writing the real number x in the form

$$x = \mu \times \beta^{e-t} \qquad \beta^{t-1} \leq \mu \leq \beta^t - 1,$$

we see that x lies between the adjacent floating point numbers $y_1 = |\mu| \ \beta^{e-t}$ and $y_2 = [\mu]\beta^{e-t}$. Thus $fl(x) = y_1$ or y_2 and we have

$$|fl(x) - x| \leq \frac{|y_2 - y_1|}{2} \leq \frac{\beta^{e-t}}{2}.$$

Hence

$$\left|\frac{fl(x) - x}{x}\right| \leq \frac{\frac{1}{2}\beta^{e-t}}{\mu \times \beta^{e-t}} \leq \frac{1}{2}\beta^{1-t} = u.$$

The last inequality is strict unless $\mu = \beta^{t-1}$, in which case $x = fl(x)$, hence the inequality of the theorem is strict. □

Theorem 2.2 says that $fl(x)$ is equal to x multiplied by a factor very close to 1. The representation $1 + \delta$ for the factor is the standard choice, but it is not the only possibility. For example, we could write the factor as e^{a}, with a bound on $|a|$ a little less than u (cf. the rp notation in §3.4).

The following modified version of this theorem can also be useful.

Theorem 2.3. *If* $x \in \mathbb{R}$ *lies in the range of* F *then*

$$fl(x) = \frac{x}{1+\delta}, \qquad |\delta| \le u.$$

Proof. See Problem 2.4. □

The widely used IEEE standard arithmetic (described in §2.3) has $\beta = 2$ and supports two precisions. Single precision has $t = 24$, $e\text{min} = -125$, $e\text{max} = 128$, and $u = 2^{-24} \approx 5.96 \times 10^{-8}$. Double precision has $t = 53$, $e\text{min} = -1021$, $e\text{max} = 1024$, and $u = 2^{-53} = 1.11 \times 10^{-16}$. IEEE arithmetic uses round to even.

It is easy to see that

$$x = \left(\beta^{t-1} + \frac{1}{2}\right) \times \beta^{e} \quad \Rightarrow \quad \left|\frac{fl(x) - x}{x}\right| \approx \frac{1}{2}\beta^{1-t},$$

$$x = \left(\beta^{t} - \frac{1}{2}\right) \times \beta^{e} \quad \Rightarrow \quad \left|\frac{fl(x) - x}{x}\right| \approx \frac{1}{2}\beta^{-t}.$$

Hence, while the relative error in representing x is bounded by $\frac{1}{2}\beta^{1-t}$ (as it must be, by Theorem 2.2), the relative error varies with x by as much as a factor β. This phenomenon is called *wobbling precision,* and is one of the reasons why small bases (in particular, $\beta = 2$) are favoured. The effect of wobbling precision is clearly displayed in Figure 2.1, which plots machine numbers x versus the relative distance from x to the next larger machine number, for $1 \le x \le 16$ in IEEE single precision arithmetic. In this plot, the relative distances range from about $2^{-23} = 1.19 \times 10^{-7}$ just to the right of a power of 2 to about $2^{-24} = 5.96 \times 10^{-8}$ just to the left of a power of 2 (see Lemma 2.1).

The notion of *ulp,* or "unit in last place", is sometimes used when describing the accuracy of a floating point result. One ulp of the normalized floating point number $y = \pm\beta^{e} \times .d_1 d_2 \ldots d_t$ is $\text{ulp}(y) = \beta^{e} \times .00 \ldots 01 = \beta^{e-t}$. If x is any real number we can say that y and x agree to $|y - x|/\text{ulp}(y)$ ulps in y. This measure of accuracy "wobbles" when y changes from a power of β to the next smaller floating point number, since $\text{ulp}(y)$ decreases by a factor β.

In MATLAB the permanent variable **eps** represents the machine epsilon (not the unit roundoff as is sometimes thought). MATLAB uses IEEE standard

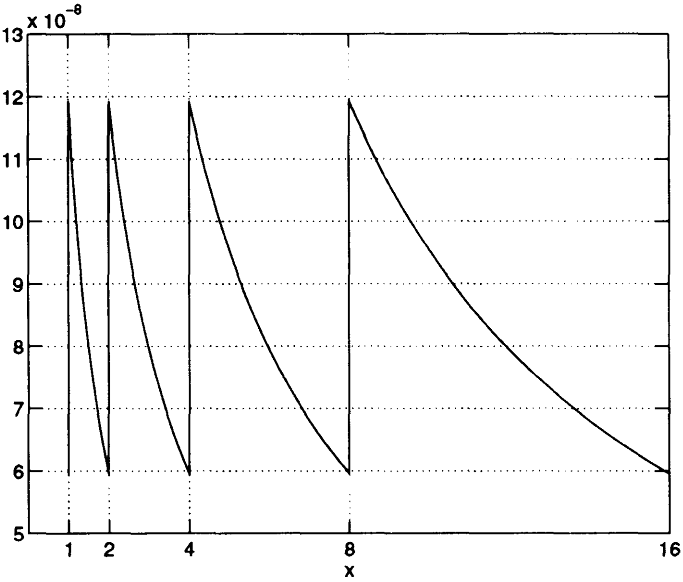


Figure 2.1. *Relative distance from* x *to the next larger machine number* ($\beta = 2$, $t = 24$), *displaying wobbling precision.*

double precision arithmetic on those machines that support it in hardware. In Fortran 90 the intrinsic function EPSILON returns the machine epsilon corresponding to the KIND of its REAL argument.

2.2. Model of Arithmetic

To carry out rounding error analysis of an algorithm we need to make some assumptions about the accuracy of the basic arithmetic operations. The most common assumptions are embodied in the following model, in which $x, y \in F$:

STANDARD MODEL

$$fl(x \operatorname{op} y) = (x \operatorname{op} y)(1+\delta), \qquad |\delta| \le u, \qquad \operatorname{op} = +, -, *, /. \tag{2.4}$$

It is normal to assume that (2.4) holds also for the square root operation.

Note that now we are using $fl(\cdot)$ with an argument that is an arithmetic expression to denote the computed value of that expression. The model says that the computed value of x op y is "as good as" the rounded exact answer, in the sense that the relative error bound is the same in both cases. However, the model does not require that $\delta = 0$ when $x \text{ op } y \in F$-a condition which obviously does hold for the rounded exact answer-so the model does not capture all the features we might require of floating point arithmetic. This model is valid for most computers, and, in particular, holds for IEEE standard arithmetic. Cases in which the model is not valid are described in §2.4.

The following modification of (2.4) can also be used (cf. Theorem 2.3):

$$\boxed{fl(x \operatorname{op} y) = \frac{x \operatorname{op} y}{1+\delta}, \qquad |\delta| \le u. \qquad (2.5)}$$

Note: Throughout this book, the standard model (2.4) is used unless otherwise stated. Most results proved using the standard model remain true with the weaker model (2.6) described below, possibly subject to slight increases in the constants. We identify problems for which the choice of model significantly affects the results that can be proved.

2.3. IEEE Arithmetic

IEEE standard 754, published in 1985 [597, 1985], defines a binary floating point arithmetic system. It is the result of several years' work by a working group of a subcommittee of the IEEE Computer Society Computer Standards Committee.

Among the design principles of the standard were that it should encourage experts to develop robust, efficient, and portable numerical programs, enable the handling of arithmetic exceptions, and provide for the development of transcendental functions and very high precision arithmetic.

The standard specifies floating point number formats, the results of the basic floating point operations and comparisons, rounding modes, floating point exceptions and their handling, and conversion between different arithmetic formats. Square root is included as a basic operation. The standard says nothing about exponentiation or transcendental functions such as exp and cos.

Two main floating point formats are defined:

Type	Size	Mantissa	Exponent	Unit roundoff	Range
Single	32 bits	23+1 bits	8 bits	$2^{-24} \approx 5.96 \times 10^{-8}$	$10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$2^{-53} \approx 1.11 \times 10^{-16}$	$10^{\pm 308}$

In both formats one bit is reserved as a sign bit. Since the floating point numbers are normalized, the most significant bit is always 1 and is not stored (except for the denormalized numbers described below). This *hidden bit* accounts for the "+l" in the table.

Table 2.2. *IEEE arithmetic exceptions and default results.*

Exception type	Example	Default result
Invalid operation	o/o, $0 \times \infty$, $\sqrt{-1}$	NaN (Not a Number)
Overflow		$\pm\infty$
Divide by zero	Finite nonzero/0	$\pm\infty$
Underflow		Subnormal numbers
Inexact	Whenever $fl(x \text{op} y) \neq x \text{op} y$	Correctly rounded result

The standard specifies that all arithmetic operations are to be performed as if they were first calculated to infinite precision and then rounded according to one of four modes. The default rounding mode is to round to the nearest representable number, with rounding to even (zero least significant bit) in the case of a tie. With this default mode, the model (2.4) is obviously satisfied. Note that computing with a single guard bit (see §2.4) will not always give the same answer as computing the exact result and then rounding. But the use of a second guard bit and a third *sticky* bit (the logical OR of all succeeding bits) enables the rounded exact result to be computed. Rounding to plus or minus infinity is also supported by the standard: this facilitates the implementation of interval arithmetic. The fourth supported mode is rounding to zero (truncation, or chopping).

IEEE arithmetic is a closed system: every arithmetic operation produces a result, whether it is mathematically expected or not, and exceptional operations raise a signal. The default results are shown in Table 2.2. The default response to an exception is to set a flag and continue, but it is also possible to take a trap (pass control to a trap handler).

A NaN is a special bit pattern that cannot be generated in the course of unexceptional operations because it has a reserved exponent field. Since the mantissa is arbitrary, subject to being nonzero, a NaN can have something about its provenance encoded in it, and this information can be used for retrospective diagnostics. A NaN is generated by operations such as 0/0, $0 \times \infty$, ∞/∞, $(+\infty)+(-\infty)$, and $\sqrt{-1}$. One creative use of the NaN is to denote uninitialized or missing data. Arithmetic operations involving a NaN return a NaN as the answer. A NaN compares as unordered and unequal with everything including itself (a NaN can be tested with the predicate $x \neq x$ or with the IEEE recommended function isnan. if provided).

The IEEE standard provides distinct representations for +0 and -0, but comparisons are defined so that +0 = -0. Signed zeros provide an elegant way to handle branch cuts in complex arithmetic; for details, see Kahan [632, 1987].

The infinity symbol is represented by a zero mantissa and the same exponent field as a NaN; the sign bit distinguishes between $\pm\infty$. The infinity symbol obeys the usual mathematical conventions regarding infinity, such as $\infty + \infty = \infty$, $(-1) \times \infty = -\infty$, and $(\text{finite})/\infty = 0$.

The standard allows subnormal numbers to be represented, instead of flushing them to zero as in many systems, and this feature permits *gradual underflow* (sometimes called *graceful underflow*). Gradual underflow makes it easier to write reliable numerical software; see Demmel [280, 1984].

The standard may be implemented in hardware or software. The first hardware implementation was the Intel 8087 floating point coprocessor, which was produced in 1981 and implements an early draft of the standard (the 8087 very nearly conforms to the present standard). This chip, together with its bigger and more recent brothers the Intel 80287, 80387, 80486 and the Pentium, is used in IBM PC compatibles (the 80486DX and Pentium are general-purpose chips that incorporate a floating point coprocessor). Other manufacturers that produce processors implementing IEEE arithmetic include DEC (Alpha), Hewlett Packard (Precision Architecture), IBM (RS/6000), Inmos (T800, T900), Motorola (680x0)) and Sun (SPARCstation).

The IEEE standard defines minimum requirements for two extended number formats: *single extended* and *double extended*. The double extended format has at least 79 bits, with at least 63 bits in the mantissa and at least 15 in the exponent; it therefore surpasses the double format in both precision and range, having unit roundoff $u \le 5.42 \times 10^{-20}$ and range at least $10^{\pm 4932}$. The purpose of the extended precision formats is not to provide for higher precision computation per se, but to enable double precision results to be computed more reliably (by avoiding intermediate overflow and underflow) and more accurately (by reducing the effect of cancellation) than would otherwise be possible. In particular, extended precision makes it easier to write accurate routines to evaluate the elementary functions, as explained by Hough [584, 1981].

A double extended format of 80 bits is supported by the Intel and Motorola chips mentioned above (which are used in many PC and Macintosh computers); these chips, in fact, normally do all their floating point arithmetic in 80 bit arithmetic (even for arguments of the single or double format). However, double extended is not supported by Sun SPARCstations or machines that use the PowerPC or DEC Alpha chips. Furthermore, the extended format (when available) is supported little by compilers and packages such as Mathematica and Maple. Kahan [636, 1994] notes that “What you do not use, you are destined to lose”, and encourages the development of benchmarks to measure accuracy and related attributes. He also explains that

> For now the 10-byte Extended format is a tolerable compromise between the value of extra-precise arithmetic and the price of implementing it to run fast; very soon two more bytes of precision

> will become tolerable, and ultimately a 16-byte format . . . That kind of gradual evolution towards wider precision was already in view when IEEE Standard 754 for Floating-Point Arithmetic was framed.

A possible side effect of the use of an extended format is the phenomenon of *double rounding,* whereby a result computed "as if to infinite precision" (as specified by the standard) is rounded first to the extended format and then to the destination format. Double rounding (which is allowed by the standard) can give a different result from that obtained by rounding directly to the destination format, and so can lead to subtle differences between the results obtained with different implementations of IEEE arithmetic (see Problems 2.9 and 3.11).

An IEEE Standard 854, which generalizes the binary standard 754, was published in 1987 [598, 1987]. It is a standard for floating point arithmetic that is independent of word length and base (although in fact only bases 2 and 10 are provided in the standard, since the drafting committee "could find no valid technical reason for allowing other radices, and found several reasons for not allowing them" [223, 1988]). Base 10 IEEE 854 arithmetic is implemented in the HP-71B calculator.

2.4. Aberrant Arithmetics

Unfortunately, not all computer floating point arithmetics adhere to the model (2.4). The most common reason for noncompliance with the model is that the arithmetic lacks a guard digit in subtraction. The role of a guard digit is easily explained with a simple example.

Consider a floating point arithmetic system with base $\beta = 2$ and $t = 3$ digits in the mantissa. Subtracting from 1.0 the next smaller floating number we have, in binary notation,

$$\begin{array}{lcl} 2^1 \times 0.100 - & \longrightarrow & 2^1 \times 0.100 - \\ 2^0 \times 0.111 & & 2^1 \times 0.0111 \\ \cline{3-3} & & 2^1 \times 0.0001 = 2^{-2} \times 0.100 \end{array}$$

Notice that to do the subtraction we had to line up the binary points, thereby unnormalizing the second number and using, temporarily, a fourth mantissa digit, known *as* a *guard digit.* Some machines do not have a guard digit. Without a guard digit in our example we would compute as follows, assuming the extra digits are simply discarded:

$$\begin{array}{lcl} 2^1 \times 0.100 - & \longrightarrow & 2^1 \times 0.100 - \\ 2^0 \times 0.111 & & 2^1 \times 0.011 \ \text{(last digit dropped)} \\ \cline{3-3} & & 2^1 \times 0.001 = 2^{-1} \times 0.100 \end{array}$$

The computed answer is too big by a factor 2 and so has relative error 1! For machines without a guard digit it is not true that

$$fl(x \pm y) = (x \pm y)(1 + \delta), \qquad |\delta| \le u,$$

but it is true that

$$fl(x \pm y) = x(1+\alpha) \pm y(1+\beta), \qquad \alpha\beta = 0, \quad |\alpha| + |\beta| \le u.$$

Our model of floating point arithmetic becomes

No GUARD DIGIT MODEL

$$fl(x \pm y) = x(1 + a) \pm y(1 + \beta), \qquad |a|, |\beta| \le u, \tag{2.6a}$$

$$fl(x \mathrm{op} y) = (x \mathrm{op} y)(1 + \delta), \qquad |\delta| \le u, \quad \mathrm{op} = *, /, \tag{2.6b}$$

where we have stated a weaker condition on a and β that is generally easier to work with.

Notable examples of machines that lack guard digits are several models of Cray computers (Cray 1, 2, X-MP, Y-MP, and C90). On these computers subtracting any power of 2 from the next smaller floating point number gives an answer that is either a factor of 2 too large (as in the example above-e.g., Cray X-MP or Y-MP) or is zero (Cray 2). In 1992 Cray announced that it would produce systems that use IEEE standard double precision arithmetic by 1995.

The lack of a guard digit is a serious drawback. It causes many algorithms that would otherwise work perfectly to fail some of the time (e.g., compensated summation-see §4.3). Here is an example of a result that holds only when a guard digit is used. This result holds for any base β.

Theorem 2.4 (Ferguson). *Let x and y be floating point numbers for which $e(x - y) \le \min(e(x), e(y))$, where e(x) denotes the exponent of x in its normalized floating point representation. If subtraction is performed with a guard digit then $x - y$ is computed exactly (assuming $x - y$ does not underflow or overflow).*

Proof. From the condition of the theorem the exponents of x and y differ by at most 1. If the exponents are the same then $fl(x - y)$ is computed exactly, so suppose the exponents differ by 1, which can happen only when x and y have the same sign. Scale and interchange x and y if necessary so that $\beta^{-1} \le y < 1 \le x < \beta$, where β is the base. Now x is represented in base β as $x_1.x_2\ldots x_t$ and the exact difference $z = x - y$ is of the form

$$\begin{array}{r} x_1.x_2 \dots x_t \;- \\ 0.y_1 \dots y_{t-1}y_t \\ \hline z_1.z_2 \dots z_t z_{t+1} \end{array}$$

But $e(x - y) \le e(y)$ and $y < 1$, so $z_1 = 0$. The algorithm for computing z forms $z_1.z_2 \dots z_t z_{t+1}$ and then rounds to t digits; since z has at most t significant digits this rounding introduces no error, and thus z is computed exactly. □

The next result is a corollary of the previous one but is more well known. It is worth stating as a separate theorem because the conditions of the theorem are so elegant and easy to check (being independent of the base), and because this weaker theorem is sufficient for many purposes.

Theorem 2.5 (Sterbenz). *Let x and y be floating point numbers with $y/2 \le x \le 2y$. If subtraction is performed with a guard digit then $x - y$ is computed exactly (assuming $x - y$ does not underflow).*

Theorem 2.5 is vital in proving that certain special algorithms work. A good example involves Heron's formula for the area A of a triangle with sides of length a, b, and c:

$$A = \sqrt{s(s-a)(s-b)(s-c)}, \qquad s = (a + b + c)/2.$$

This formula is inaccurate for needle-shaped triangles: if $a \approx b + c$ then $s \approx a$ and the term $s - a$ suffers severe cancellation. A way around this difficulty, devised by Kahan, is to rename a, b, and c so that $a \ge b \ge c$ and then evaluate

$$A = \frac{1}{4}\sqrt{\big(a+(b+c)\big)\big(c-(a-b)\big)\big(c+(a-b)\big)\big(a+(b-c)\big)}. \tag{2.7}$$

The parentheses are essential! Kahan has shown that this formula gives the area with a relative error bounded by a modest multiple of the unit roundoff *provided* that a guard digit is used in subtraction [457, 1991, Thm. 3], [634, 1990] (see Problem 2.22). If there is no guard digit, the computed result can be very inaccurate.

Kahan has made these interesting historical comments about guard digits [634, 1990]:

> CRAYs are not the first machines to compute differences blighted by lack of a guard digit. The earliest IBM '360s, from 1964 to 1967, subtracted and multiplied without a hexadecimal guard digit until SHARE, the IBM mainframe user group, discovered why the consequential anomalies were intolerable and so compelled a guard digit to be retrofitted. The earliest Hewlett-Packard financial calculator, the HP-80, had a similar problem. Even now, many a calculator (but not Hewlett-Packard's) lacks a guard digit.

2.5. Choice of Base and Distribution of Numbers

What base β is best for a floating point number system? Most modern computers use base 2. Most hand-held calculators use base 10, since it makes the calculator easier for the user to understand (how would you explain to a naive user that 0.1 is not exactly representable on a base 2 calculator?). IBM mainframes traditionally have used base 16. Even base 3 has been tried-in an experimental machine called SETUN, built at Moscow State University in the late 1950s [1066, 1960].

Several criteria can be used to guide the choice of base. One is the impact of wobbling precision: as we saw at the end of §2.1, the spread of representation errors is smallest for small bases. Another possibility is to measure the worst-case representation error or the mean square representation error. The latter quantity depends on the assumed distribution of the numbers that are represented. Brent [144, 1973] shows that for the logarithmic distribution the worst-case error and the mean square error are both minimal for (normalized) base 2, provided that the most significant bit is not stored explicitly.

The logarithmic distribution is defined by the property that the proportion of base β numbers with leading significant digit n is

$$\log_\beta(n+1) - \log_\beta n = \log_\beta\Bigl(1 + \frac{1}{n}\Bigr).$$

It appears that in practice real numbers *are* logarithmically distributed. In 1938, Benford [90, 1938] noticed, as had Newcomb [794, 1881] before him, that the early pages of logarithm tables showed greater signs of wear than the later ones. (He was studying dirty books!) This prompted him to carry out a survey of 20,229 “real-life” numbers, whose decimal representations he found matched the logarithmic distribution closely.

The observed logarithmic distribution of leading significant digits has not been fully explained. Some proposed explanations are based on the assumption that the actual distribution is scale invariant, but this assumption is equivalent to the observation to be explained [1032, 1984]. Barlow [57, 1981], [58, 1981], [60, 1988] and Turner [1031, 1982], [1032, 1984] give useful insight by showing that if uniformly distributed numbers are multiplied together, then the resulting distribution converges to the logarithmic one; see also Boyle [141, 1994]. Furthermore, it is an interesting result that the leading significant digits of the numbers q^k, $k = 0,1,2,\ldots$, are logarithmically distributed if q is positive and is not a rational power of 10; when $q = 2$ and the digit is 7 this is Gelfand’s problem [829, 1981, pp. 50-51].

The nature of the logarithmic distribution is striking. For decimal numbers, the digits 1 to 9 are not equally likely to be a leading significant digit. The probabilities are as follows:

1	2	3	4	5	6	7	8	9
0.301	0.176	0.125	0.097	0.079	0.067	0.058	0.051	0.046

As an example, here is the leading significant digit distribution for the elements of the inverse of one random 100 × 100 matrix from the normal $N(0, 1)$ distribution:

1	2	3	4	5	6	7	8	9
0.334	0.163	0.100	0.087	0.077	0.070	0.063	0.056	0.051

For an entertaining survey of work on the distribution of leading significant digits see Raimi [856, 1976] (and also the popular article [855, 1969]).

2.6. Statistical Distribution of Rounding Errors

Most rounding error analyses, including all the ones in this book, are designed to produce worst-case bounds for the error. The analyses ignore the signs of rounding errors and are often the result of many applications of the triangle inequality and the submultiplicative inequality. Consequently, although the bounds may well give much insight into a method, they tend to be pessimistic if regarded as error estimates.

Statistical statements about the effect of rounding on a numerical process can be obtained from statistical analysis coupled with probabilistic models of the rounding errors. For example, a well-known rule of thumb is that a more realistic error estimate for a numerical met hod is obtained by replacing the dimension-dependent constants in a rounding error bound by their square root: thus if the bound is $f(n)u$, the rule of thumb says that the error is typically of order $\sqrt{f(n)}u$ (see, for example, Wilkinson [1088, 1963, pp. 26, 102]). This rule of thumb can be supported by assuming that the rounding errors are independent random variables and applying the central limit theorem. Statistical analysis of rounding errors goes back to one of the first papers on rounding error analysis, by Goldstine and von Neumann [462, 1951].

As we noted in §1.17, rounding errors are *not* random. See Problem 2.10 for an example of how two rounding errors cancel in one particular class of computations. Forsythe [389, 1959] points out that rounding errors do not necessarily behave like independent random variables and proposes a random form of rounding (intended for computer testing) to which statistical analysis is applicable.

Henrici [517, 1962], [518, 1963], [519, 1964] assumes models for rounding errors and then derives the probability distribution of the overall error, mainly in the context of difference methods for differential equations. Hull and Swenson [593, 1966] give an insightful discussion of probabilistic models, pointing out that "There is no claim that ordinary rounding and chopping are random processes, or that successive errors are independent. The question to be decided is whether or not these particular probabilistic models of the processes will adequately describe what actually happens" (see also the ensuing note by Henrici [520, 1966]).

Since the late 1980s Chaitin-Chatelin and her co-workers have been developing a method called PRECISE, which involves a statistical analysis of the effect on a computed solution of random perturbations in the data; see Brunet [152, 1989], Chatelin and Brunet [202, 1990], and Chaitin-Chatelin and Frayssé [190, 1996]. This approach is superficially similar to the earlier CESTAC (permutation-perturbation) method of La Porte and Vignes [153, 1986], [682, 1974], [1054, 1986], but differs from it in several respects. CESTAC deals with the arithmetic reliability of algorithms, whereas PRECISE is designed as a tool to explore the robustness of numerical algorithms as a function of parameters such as mesh size, time step, and nonnormality.

Several authors have investigated the distribution of rounding errors under the assumption that the mantissas of the operands are from a logarithmic distribution, and for different modes of rounding; see Barlow and Bareiss [62, 1985] and the references therein.

Other work concerned with statistical modelling of rounding errors includes that of Tienari [1002, 1970] and Linnainmaa [704, 1975].

2.7. Alternative Number Systems

The floating point format is not the only means for representing numbers in finite precision arithmetic. Various alternatives have been proposed, though none has achieved widespread use.

A particularly elegant system is the "level index arithmetic" of Clenshaw, Olver, and Turner, in which a number $x > 1$ is represented by $\xi = l + f$, where $f \in [0,1]$ and

$$x = e^{e^{\cdot^{\cdot^{\cdot^{e^f}}}}}, \qquad \text{or} \qquad f = \ln(\ln(\dots(\ln x)\dots)),$$

where the exponentiation or logarithm is performed l times (l is the "level"). If $0 < x < 1$, then x is represented by the reciprocal of the representation for $1/x$. An obvious feature of the level index system is that it can represent much larger and smaller numbers than the floating point system, for similar word lengths. A price to be paid is that addition and subtraction are more complicated (and more costly) than in floating point arithmetic. For very readable introductions to level index arithmetic see Clenshaw and Olver [213, 1984] and Turner [1033, 1991], and for more details see Clenshaw, Olver, and Turner [214, 1989]. Level index arithmetic is somewhat controversial in that there is disagreement about its advantages and disadvantages with respect to floating point arithmetic; see Demmel [282, 1987]. A number system involving levels has also been proposed by Matsui and Iri [736, 1981]; in their system,

Table 2.3. *Test arithmetics.*

Hardware	Software	$\lvert 3 \times (4/3 - 1) - 1 \rvert$[a]
Casio fx-140 (≈1979)		1×10^{-9}
Casio fx-992VB (≈1990)		1×10^{-13}
HP 48G (1993)		1×10^{-11}
Sharp EL-5020 (1994)		0.0[b]
486DX	MATLAB 4.2 (1994)	$2.2\ldots \times 10^{-16}$
486DX	WATFOR-77[c] V3.0 (1988)	$2.2\ldots \times 10^{-16}$
486DX	FTN 90[d] (1993)	$2.2\ldots \times 10^{-16}$
486DX	MS-DOS QBasic 1.1	$1.1\ldots \times 10^{-19}$ [e]

---

[a] Integers in the test expression are typed as real constants 3.0, etc., for the Fortran tests.

[b] 1×10^{-9} if 4/3 is stored and recalled from memory.

[c] WATCOM Systems Inc.

[d] Salford Software/Numerical Algorithms Group, Version 1.2.

[e] $2.2\ldots \times 10^{-16}$ if 4/3 is stored and recalled from a variable.

the number of bits allocated to the mantissa and exponent is allowed to vary (within a fixed word size).

Other number systems include those of Swartzlander and Alexopolous [981, 1975], Matula and Kornerup [741, 1985], and Hamada [495, 1987]. For summaries of alternatives to floating point arithmetic see the section "Alternatives to Floating-Point Some Candidates" in [214, 1989], and Knuth [668, 1981, Chap. 4].

2.8. Accuracy Tests

How can you test the accuracy of the floating point arithmetic on a computer or pocket calculator? There is no easy way, though a few software packages are available to help with the tasks in specific programming languages (see §25.6). There are, however, a few quick and easy tests that may reveal weaknesses. The following list is far from comprehensive and good performance on these tests does not imply that an arithmetic is correct. Results from the tests are given in Tables 2.4-2.5 for the selected floating point arithmetics described in Table 2.3. Double precision was used for the compiled languages. The last column of Table 2.3 gives an estimate of the unit roundoff (see Problem 2.14). The estimate produced by QBasic indicates that the compiler used extended precision in evaluating the estimate.

1. (Cody [221, 1982]) Evaluate $\sin(22) = -8.8513\ 0929\ 0403\ 8759\ 2169 \times 10^{-3}$ (shown correct to 21 digits). This is a difficult test for the range

Table 2.4. Sine *test.*

Machine	sin(22)
Exact	$-8.8513\ 0929\ 0403\ 8759 \times 10^{-3}$
Casio fx-140	$-8.8513\ 62 \times 10^{-3}$
Casio fx-992VB	$-8.8513\ 0929\ 096 \times 10^{-3}$
HP 48G	$-8.8513\ 0929\ 040 \times 10^{-3}$
Sharp EL-5020	$-8.8513\ 0915\ 4 \times 10^{-3}$
MATLAB 4.2	$-8.8513\ 0929\ 0403\ 876 \times 10^{-3}$
WATFOR-77	$-8.8513\ 0929\ 0403\ 880 \times 10^{-3}$
FTN 90	$-8.8513\ 0929\ 0403\ 876 \times 10^{-3}$
QBasic	$-8.8513\ 0929\ 0403\ 876 \times 10^{-3}$

Table 2.5. *Exponentation test. No entry for last column means same value as previous column.*

Machine	2.5^{125}	exp(125log(2.5))
Exact	$5.5271\ 4787\ 5260\ 4446 \times 10^{49}$	$5.5271\ 4787\ 5260\ 4446 \times 10^{49}$
Casio fx-140	$5.5271\ 477 \times 10^{49}$	$5.5271\ 463 \times 10^{49}$
Casio fx-992VB	$5.5271\ 4787\ 526 \times 10^{49}$	
HP 48G	$5.5271\ 4787\ 526 \times 10^{49}$	$5.5271\ 4787\ 377 \times 10^{49}$
Sharp EL-5020	5.5271 4787 3 × 1049	$5.5271\ 4796\ 2 \times 10^{49}$
MATLAB 4.2	$5.5271\ 4787\ 5260\ 445 \times 10^{49}$	$5.5271\ 4787\ 5260\ 459 \times 10^{49}$
WATFOR-77	$5.5271\ 4787\ 5260\ 450 \times 10^{49}$	$5.5271\ 4787\ 5260\ 460 \times 10^{49}$
FTN 90	$5.5271\ 4787\ 5260\ 445 \times 10^{49}$	$5.5271\ 4787\ 5260\ 459 \times 10^{49}$
QBasic	$5.5271\ 4787\ 5260\ 444 \times 10^{49}$	

reduction used in the sine evaluation (which brings the argument within the range $[-\pi/2, \pi/2]$, and which necessarily uses an approximate value of π), since 22 is close to an integer multiple of π.

2. (Cody [221, 1982]) Evaluate $2.5^{125} = 5.5271\ 4787\ 5260\ 4445\ 6025 \times 10^{49}$ (shown correct to 21 digits). One way to evaluate $z = x^y$ is as $z = \exp(y \log x)$. But to obtain z correct to within a few ulps it is not sufficient to compute exp and log correct to within a few ulps; in other words, the composition of two functions evaluated to high accuracy is not necessarily obtained to the same accuracy. To examine this particular case, write

$$w := y \log x, \qquad z = \exp(w).$$

If $w \to w+\Delta w$ then $z \to z+\Delta z$, where $z+\Delta z = \exp(w+\Delta w) = \exp(w)\exp(\Delta w) \approx \exp(w)(1+\Delta w)$, so $\Delta z/z \approx \Delta w$. In other words, the relative error of z depends on the absolute error of w and hence on the size of w. To obtain z correct to within a few ulps it is necessary to use extra precision in calculating the logarithm and exponential [228, 1980, Chap. 7].

3. (Karpinski [645, 1985]) A simple test for the presence of a guard digit on a pocket calculator is to evaluate the expressions

$$9/27 * 3 - 1, \qquad 9/27 * 3 - 0.5 - 0.5,$$

which are given in a form that can be typed directly into most four-function calculators. If the results are equal then a guard digit is present. Otherwise there is probably no guard digit (we cannot be completely sure from this simple test). To test for a guard digit on a computer it is best to run one of the diagnostic codes described in §25.5.

2.9. Notes and References

The classic reference on floating point arithmetic, and on all aspects of rounding error analysis, is Wilkinson's *Rounding Errors in Algebraic Processes* (REAP) [1088, 1963]. Wilkinson was uniquely qualified to write such a book, for not only was he the leading expert in rounding error analysis, but he was one of the architects and builders of the Automatic Computing Engine (ACE) at the National Physical Laboratory [1082, 1954]. The Pilot (prototype) ACE first operated in May 1950, and an engineered version was later sold commercially as the DEUCE Computer by the English Electric Company. Wilkinson and his colleagues were probably the first to write subroutines for floating point arithmetic. and this enabled them to accumulate practical experience of floating point arithmetic much earlier than anyone else [357, 1976], [1099, 1980].

In REAP, Wilkinson gives equal consideration to fixed point and floating point arithmetic. In fixed point arithmetic, all numbers are constrained to lie in a range such as [-1,1], as if the exponent were frozen in the floating point representation (2.1). Preliminary analysis and the introduction of scale factors during the computation is needed to keep numbers in the permitted range. We consider only floating point arithmetic in this book. REAP, together with Wilkinson's second book, *The Algebraic Eigenvalue Problem* (AEP) [1089, 1965], has been immensely influential in the areas of floating point arithmetic and rounding error analysis.

Wilkinson's books were preceded by the paper *Error Analysis of Floating-Point Computation* [1084, 1960], in which he presents the model (2.4) for floating point arithmetic and applies the model to several algorithms for the eigenvalue problem. This paper has hardly dated and is still well worth reading.

Another classic book devoted entirely to floating point arithmetic is Sterbenz's *Floating-Point Computation* [938, 1974]. It contains a thorough treatment of low-level details of floating point arithmetic, with particular reference to IBM 360 and IBM 7090 machines. It also contains a good chapter on rounding error analysis and an interesting collection of exercises. R. W. Hamming has said of this book, "Nobody should ever have to know that much about floating-point arithmetic. But I'm afraid sometimes you might" [833, 1988]. Although Sterbenz's book is now dated in some respects, it remains a useful reference.

A third important reference on floating point arithmetic is Knuth's *Seminumerical Algorithms* [668, 1981, §4.2], from his *Art of Computer Programming* series. Knuth's lucid presentation includes historical comments and challenging exercises (with solutions).

The first analysis of floating point arithmetic was given by Samelson and Bauer [890, 1953]. Later in the same decade Carr [187, 1959] gave a detailed discussion of error bounds for the basic arithmetic operations.

An up-to-date and very readable reference on floating point arithmetic is the survey paper by Goldberg [457, 1991], which includes a detailed discussion of IEEE arithmetic. A less mathematical, more hardware-oriented discussion can be found in the appendix "Computer Arithmetic" written by Goldberg that appears in the book on computer architecture by Hennessy and Patterson [515, 1990].

A fascinating historical perspective on the development of computer floating point arithmetics, including background to the development of the IEEE standard, can be found in the textbook by Patterson and Hennessy [822, 1994, §4.11].

The idea of representing floating point numbers in the form (2.1) is found, for example, in the work of Forsythe [393, 1969], Matula [740, 1970], and Dekker [275, 1971].

An alternative definition of $fl(x)$ is the nearest $y \in G$ satisfying $|y| \leq |x|$. This operation is called *chopping*, and does not satisfy our definition of rounding. Chopped arithmetic is used in the IBM/370 floating point system.

The difference between chopping and rounding (to nearest) is highlighted by a discrepancy in the index of the Vancouver Stock Exchange in the early 1980s [852, 1983]. The exchange established an index in January 1982, with the initial value of 1000. By November 1983 the index had been hitting lows in the 520s, despite the exchange apparently performing well. The index was recorded to three decimal places and it was discovered that the computer program calculating the index was chopping instead of rounding to produce the final value. Since the index was recalculated thousands of times a day, each time with a nonpositive final error, the bias introduced by chopping became significant. Upon recalculation with rounding the index almost doubled!

When there is a tie in rounding, two possible strategies are to round to the number with an even last digit and to round to the one with an odd last digit. Both are stable forms of rounding in the sense that

$$fl(((x+y)-y)+y)-y) = fl((x+y)-y),$$

as shown by Reiser and Knuth [869, 1975], [668, 1981, p. 222]. For other rules, such as round away from zero, repeated subtraction and addition of the same number can yield an increasing sequence, a phenomenon known as *drift*. For bases 2 and 10 rounding to even is preferred to rounding to odd. After rounding to even a subsequent rounding to one less place does not involve a tie. Thus we have the rounding sequence 2.445, 2.44, 2.4 with round to even, but 2.445, 2.45. 2.5 with round to odd. For base 2, round to even causes computations to produce integers more often [640, 1979] as a consequence of producing a zero least significant bit. Rounding to even in the case of ties seems to have first been suggested by Scarborough in the first edition (1930) of [897, 1950].

Predictions based on the growth in the size of mathematical models solved as the memory and speed of computers increase suggest that floating point arithmetic with unit roundoff $u \approx 10^{-32}$ will be needed for some applications on future supercomputers [48, 1989].

The model (2.4) does not fully describe any floating point arithmetic. It is merely a tool for error analysis-one that has been remarkably successful in view of our current understanding of the numerical behaviour of algorithms. There have been various attempts to devise formal models of floating point arithmetic, by specifying sets of axioms in terms of which error analysis can be performed. Some attempts are discussed in §25.7.4. No model yet proposed has been truly successful. Priest [844, 1992] conjectures that the task of "encapsulating all that we wish to know about floating point arithmetic in a single set of axioms" is impossible, and he gives some motivation for this conjecture.

Under the model (2.4), floating point arithmetic is not associative with respect to any of the four basic operations: $(a \boxed{\text{op}} b) \boxed{\text{op}} c \neq a \boxed{\text{op}} (b \boxed{\text{op}} c)$, op $= +,-,*,/$, where $a \boxed{\text{op}} b := fl(a \text{ op } b)$. Nevertheless, floating point arithmetic enjoys some algebraic structure, and it is possible to carry out error analysis in the " $\boxed{\text{op}}$ algebra". Fortunately, it was recognized by Wilkinson and others in the 1950s that this laboured approach is unnecessarily complicated, and that it is much better to work with the exact equations satisfied by the computed quantities. As Parlett [821, 1990] notes, though, "There have appeared a number of ponderous tomes that do manage to abstract the computer's numbers into a formal structure and burden us with more jargon."

A draft proposal of IEEE Standard 754 is defined and described in [599, 1981]. That article, together with others in the same issue of the journal

Computer, provides a very readable description of IEEE arithmetic. In particular, an excellent discussion of gradual underflow is given by Coonen [243, 1981]. A draft proposal of IEEE Standard 854 is presented, with discussion, in [225, 1984].

W. M. Kahan of the University of California at Berkeley received the 1989 ACM Turing Award for his contributions to computer architecture and numerical analysis, and in particular for his work on IEEE floating point arithmetic standards 754 and 854.

An interesting examination of the implications of the IEEE standard for high-level languages such as Fortran is given by Fateman [365, 1982]. Topics discussed include trap handling and how to exploit NaNs. For an overview of hardware implementations of IEEE arithmetic, and software support for it, see Cody [223, 1988].

Producing a fast and correct implementation of IEEE arithmetic is a difficult task. Correctness is especially important for a microprocessor (as opposed to a software) implementation, because of the logistical difficulties of correcting errors when they are found. In late 1994, much publicity was generated by the discovery of a flaw in the floating point divide instruction of Intel's Pentium chip. Because of some missing entries in a lookup table on the chip, the FPDIV instruction could give as few as four correct significant decimal digits for double precision floating point arguments with certain special bit patterns [916, 1994]. The flaw had been discovered by Intel in the summer of 1994 during ongoing testing of the Pentium processor, but it had not been publically announced. In October 1994, a mathematician doing research into prime numbers independently discovered the flaw and reported it to the user community. Largely because of the way in which Intel responded to the discovery of the flaw, the story was reported in national newspapers (e.g., the *New York Times* [727, 1994]) and generated voluminous discussion on Internet newsgroups (notably comp. sys. intel). Intel corrected the bug in 1994 and, several weeks after the bug was first reported, offered to replace faulty chips. For a very readable account of the Pentium FPDIV bug story, see Moler [772, 1995]. To emphasize that bugs in implementations of floating point arithmetic are far from rare, we mention that the Calculator application in Microsoft Windows 3.1 evaluates $fl(2.01 - 2.00) = 0.0$.

Computer chip designs can be tested in two main ways: by software simulations and by applying formal verification techniques. Formal verification aims to prove mathematically that the chip design is correct, and this approach is now being used by Intel and other chip manufacturers [452, 1995]. The implementation of IEEE arithmetic for the Inmos T800 transputer in the 1980s was done with the help of formal methods. The IEEE standard was translated into the set-theoretic specification language Z, and then Occam procedures were written that were proved to adhere to the specifications. For details, see Barrett [69, 1989] or, for a more informal overview, Shepherd and

Wilson [917, 1989].

The floating point operation op (op = +,-,*, or /) is *monotonic* if $fl(a \text{ op } b) \le fl(c \text{ op } d)$ whenever a, b, c, and d are floating point numbers for which $a \text{ op } b \le c \text{ op } d$ and neither $fl(a \text{ op } b)$ nor $fl(c \text{ op } d)$ overflows. IEEE arithmetic is monotonic, as is any correctly rounded arithmetic. Monotonic arithmetic is important in the bisection algorithm for finding the eigenvalues of a symmetric tridiagonal matrix; see Demmel, Dhillon, and Ren [289, 1994], who give rigorous correctness proofs of some bisection implementations in floating point arithmetic. Ferguson and Brightman [371, 1991] derive conditions that ensure that an approximation to a monotonic function preserves the monotonicity on a set of floating point numbers.

On computers of the 1950s (fixed point) multiplication was slower than (fixed point) addition by up to an order of magnitude [693, 1980, Apps. 2, 3]. For most modern computers it is a rule of thumb that a floating point addition and multiplication take about the same amount of time, while a floating point division is 2-10 times slower, and a square root operation (in hardware) is 1-2 times slower than a division.

Some computers have the ability to perform a floating point multiplication followed by an addition or subtraction, $x * y + z$ or $x * y - z$, as though it were a single floating point operation. For example, the IBM RISC System/6000 has a fused multiply-add (FMA) operation that forms $x * y + z$ with just a single rounding error, at the end, the multiplication and addition being performed at twice the precision of the operands (and by overlapping additions and multiplications the RS/6000 can perform a sequence of FMAs in one cycle each) [596, 1993]. For a clever use of an FMA operation to achieve increased accuracy in a computation, see Problem 2.25.

During the design of the IBM 7030, Sweeney [982, 1965] collected statistics on the floating point additions carried out by selected application programs on an IBM 704. He found that 11% of all instructions traced were floating point additions. Details were recorded of the shifting needed to align floating point numbers prior to addition, and the results were used in the design of the shifter on the IBM 7030.

The word *bit,* meaning binary digit, first appeared in print in a 1948 paper of Claude E. Shannon, but the term was apparently coined by John W. Tukey [1022, 1984]. The word *byte,* meaning a group of (usually eight) bits, did not appear in print until 1959 [156, 1981].

The earliest reference we know for Theorem 2.5 is Sterbenz [938, 1974, Thm. 4.3.1]. Theorem 2.4 is due to Ferguson [370, 1995], who proves a more general version of the theorem that allows for trailing zero digits in x and y. A variation in which the condition is $0 \le y \le x \le y + \beta^e$. where $e = \min\{j: \beta^j \ge y\}$, is stated by Ziv [1132, 1991] and can be proved in a similar way.

For more on the choice of base, see Cody [227, 1973] and Kuki and Cody [677, 1973]. Buchholz's paper *Fingers or Fists*? [155, 1959] on binary versus decimal representation of numbers on a computer deserves mention for its clever title, though the content is only of historical interest.

The model (2.4) ignores the possibility of underflow and overflow. To take underflow into account the model must be modified to

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta) + h, \qquad \text{op} = +,-,*,/. \tag{2.8}$$

As before, $|\delta| \le u$. If underflow is gradual, as in IEEE arithmetic, then $|h| \le \frac{1}{2}\beta^{e_{\min}-t} = \lambda u$, which is half the spacing between the subnormal numbers ($\lambda = \beta^{e_{\min}-1}$ is the smallest positive normalized floating point number); if underflows are flushed to zero then $|h| \le \lambda$. Only one of δ and h is nonzero: δ if no underflow occurs, otherwise h. With gradual underflow the *absolute* error of an underflowed result is no greater than the smallest (bound for the) absolute error that arises from an operation $fl(x \text{ op } y)$ in which the arguments and result are normalized. For more details, and a thorough discussion of how error analysis of standard algorithms is affected by using the model (2.8), see the perceptive paper by Demmel [280, 1984]. Another relevant reference is Neumaier [789, 1985].

Algorithms for evaluating elementary functions in IEEE arithmetic are developed by Tang [987, 1989], [989, 1990], [991, 1992], Gal and Bachelis [412, 1991], and Ziv [1132, 1991]. Tang [990, 1991] gives a very readable description of table lookup algorithms for evaluating elementary functions, which are used in a number of current chips.

Algorithms for evaluating complex elementary functions that exploit exception handling and assume the availability of algorithms for the real elementary functions are presented by Hull, Fairgrieve, and Tang [592, 1994]. For details of how elementary functions are evaluated on many of today's pocket calculators see Schelin [898, 1983].

An important problem not considered in this chapter is the conversion of numbers between decimal and binary representations. These conversions are needed whenever numbers are read into a computer or printed out. They tend to be taken for granted, but if not done carefully they can lead to puzzling behaviour, such as a number read in as 0.1 being printed out as 0.099. . .9. To be precise, the problems of interest are (a) convert a number represented in decimal notation to the best binary floating point representation of a given precision, and (b) given a binary floating point number, print a correctly rounded decimal representation, either to a given number of significant digits or to the smallest number of significant digits that allows the number to be re-read without loss of accuracy. Algorithms for solving these problems are given by Clinger [220, 1990] and Steele and White [936, 1990]; Gay [431, 1990] gives some improvements to the algorithms and C code implementing them. Precise requirements for binary-decimal conversion are specified in the IEEE arithmetic standard. A program for testing the correctness of binary-decimal conversion routines is described by Paxson [823, 1991]. Early references on base conversion are Goldberg [458, 1967] and Matula [739, 1968], [740, 1970]. It is interesting to note that, in Fortran or C, where the output format for a "print" statement can be precisely specified, most compilers will, for an (in)appropriate choice of format, print a decimal string that contains many more significant digits than are determined by the floating point number whose value is being represented.

Other authors who have analysed various aspects of floating (and fixed) point arithmetic include Diamond [305, 1978], Urabe [1037, 1968], and Feldstein, Goodman, and co-authors [471, 1975], [368, 1982], [472, 1985], [369, 1986]. For a survey of computer arithmetic up until 1976 that includes a number of references not given here, see Garner [420, 1976].

Problems

> *The exercise had warmed my blood, and*
> *I was beginning to enjoy myself amazingly.*
> -JOHN BUCHAN, *The Thirty-Nine Steps* (1915)

2.1. How many normalized numbers and how many subnormal numbers are there in the system F defined in (2.1) with $e_{\min} \le e \le e_{\max}$? What are the figures for IEEE single and double precision (base 2)?

2.2. Prove Lemma 2.1.

2.3. In IEEE arithmetic how many double precision numbers are there between any two adjacent nonzero single precision numbers?

2.4. Prove Theorem 2.3.

2.5. Show that

$$0.1 = \sum_{i=1}^{\infty}(2^{-4i} + 2^{-4i-1})$$

and deduce that 0.1 has the base 2 representation $0.0001\overline{1100}$ (repeating last 4 bits). Let $\widehat{x} = fl(0.1)$ be the rounded version of 0.1 obtained in binary IEEE single precision arithmetic ($u = 2^{-24}$). Show that $(x - \widehat{x})/x = -\frac{1}{4}u$.

2.6. What is the largest integer p such that all integers in the interval $[-p,p]$ are exactly representable in IEEE double precision arithmetic? What is the corresponding p for IEEE single precision arithmetic?

2.7. Which of the following statements is true in IEEE arithmetic, assuming that a and b are normalized floating point numbers and that no exception occurs in the stated operations?

1. $fl(a \text{ op } b) = fl(b \text{ op } a)$, op $= +,*$.
2. $fl(b - a) = -fl(a - b)$.

3. $fl(a + a) = fl(2 * a)$.
4. $fl(0.5 * a) = fl(a/2)$.
5. $fl((a + b) + c) = fl(a + (b + c))$.
6. $a \le fl((a + b)/2) \le b$, given that $a \le b$.

2.8. Show that the inequalities $a \le fl((a + b)/2) \le b$, where a and b are floating point numbers with $a \le b$, can be violated in base 10 arithmetic. Show that $a \le fl(a+(b-a)/2) \le b$ in base β arithmetic, for any β, assuming the use of a guard digit.

2.9. What is the result of the computation $\sqrt{1 - 2^{-53}}$ in IEEE double precision arithmetic, with and without double rounding from an extended format with a 64-bit mantissa?

2.10. A theorem of Kahan [457, 1991, Thm. 7] says that if $\beta = 2$ and the arithmetic rounds as specified in the IEEE standard, then for integers m and n with $|m| < 2^{t-1}$ and $n = 2^i + 2^j$ (some i, j), $fl((m/n)*n) = m$. Thus, for example, $fl((1/3) * 3) = 1$ (even though $fl(1/3) \ne 1/3$). The sequence of allowable n begins 1,2,3,4,5,6,8,9,10,12,16,17,18,20, so Kahan's theorem covers many common cases. Test the theorem on your computer.

2.11. Investigate the leading significant digit distribution for numbers obtained as follows.

1. k^n, $n = 0{:}1000$ for $k = 2$ and 3.
2. $n!$, $n = 1{:}1000$.
3. The eigenvalues of a random symmetric matrix.
4. Physical constants from published tables.
5. From the front page of the *London Times* or the *New York Times*.

(Note that in writing a program for the first case you can form the powers of 2 or 3 in order, following each multiplication by a division by 10, as necessary, to keep the result in the range [1,10]. Similarly for the second case.)

2.12. (Edelman [343, 1994]) Let x be a floating point number in IEEE double precision arithmetic satisfying $1 \le x < 2$. Show that $fl(x*(1/x))$ is either 1 or $1 - \epsilon/2$, where $\epsilon = 2^{-52}$ (the machine epsilon).

2.13. (Edelman [343, 1994]) Consider IEEE double precision arithmetic. Find the smallest positive integer j such that $fl(x*(1/x)) \ne 1$, where $x = 1 + j\epsilon$, with $\epsilon := 2^{-52}$ (the machine epsilon).

2.14. Kahan has stated that "an (over-)estimate of u can be obtained for almost any machine by computing $|3 \times (4/3 - 1) - 1|$ using rounded floating-point for every operation". Test this estimate against u on any machines available to you.

2.15. What is 0^0 in IEEE arithmetic?

2.16. Evaluate these expressions in any IEEE arithmetic environment available to you. Are the values returned what you would expect? (None of the results is specified by the IEEE standard.)

1. 1^∞.
2. 2^∞.
3. $\exp(\infty)$, $\exp(-\infty)$.
4. $\mathrm{sign}(\mathrm{NaN})$, $\mathrm{sign}(-\mathrm{NaN})$.
5. NaN^0.
6. ∞^0.
7. 1^{NaN}.
8. $\log(\infty)$, $\log(-\infty)$, $\log(0)$.

2.17. In the course of solving $ax^2 - 2bx + c = 0$ for x, the expression $\sqrt{b^2 - ac}$ must be computed. Can the true value of $b^2 - ac$ be nonnegative and yet its computed value be negative?

2.18. Can Theorem 2.4 be strengthened to say that $fl(x - y)$ is computed exactly whenever the exponents of $x \geq 0$ and $y \geq 0$ differ by at most 1?

2.19. Two requirements that we might ask of a routine for computing $\sqrt{x}$ in floating point arithmetic are that the identities $\sqrt{x^2} = |x|$ and $(\sqrt{x})^2 = |x|$ be satisfied. Which, if either, of these is a reasonable requirement?

2.20. Are there any floating point values of x and y (excepting values both 0, or so huge or tiny to cause overflow or underflow) for which the computed value of $x/\sqrt{x^2 + y^2}$ exceeds 1?

2.21. (Kahan) A natural way to compute the maximum of two numbers x and y is with the code

```
% max(x, y)
if $x > y$ then
    max = $x$
else
    max = $y$
end
```

Does this code always produce the expected answer in IEEE arithmetic?

2.22. Prove that Kahan's formula (2.7) computes the area of a triangle accurately if a guard digit is used in subtraction. (Hint: you will need one invocation of Theorem 2.5.)

2.23. (Kahan) Describe the result of the computation $y = (x + x) - x$ on a binary machine with a guard digit and one without a guard digit.

2.24. (Kahan) Let $f(x) = (((x - 0.5) + x) - 0.5) + x$. Show that if f is evaluated as shown in single or double precision binary IEEE arithmetic then $f(x) \neq 0$ for all floating point numbers x.

2.25. (Kahan) Consider a machine that can perform a fused multiply-add operation with just a single rounding error:

$$fl(x + y * z) = (x + y * z)(1 + \delta), \qquad |\delta| \leq u.$$

Show that, on such a machine, the algorithm

$$\begin{aligned} &w = bc \\ &e = w - b * c \\ &x = (a * d - w) + e \end{aligned}$$

computes $x = \det(\begin{bmatrix} a & b \\ c & d \end{bmatrix})$ with high relative accuracy.

2.26. Derive Newton's method for solving $f(x) = a - 1/x = 0$. This method was used on early computers (and is still used on some Cray computers, for example) to implement reciprocation in terms of multiplication and thence division as $a/b = a * (1/b)$; see, e.g., [506, 1946].

2.27. Suppose we have an iterative algorithm for computing $z = x/y$. Derive a convergence test that terminates the iteration (only) when full accuracy has been achieved. Assume the use of IEEE arithmetic with gradual underflow (use (2.8)).

Chapter 3
Basics

A method of inverting the problem of round-off error is proposed which we plan to employ in other contexts and which suggests that it may be unwise to separate the estimation of round-off error from that due to observation and truncation.

-WALLACE J. GIVENS, *Numerical Computation of the Characteristic Values of a Real Symmetric Matrix* (1954)

The enjoyment of one's tools is an essential ingredient of successful work.

-DONALD E. KNUTH, *The Art of Computer Programming, Volume 2, Seminumerical Algorithms* (1981)

The subject of propagation of rounding error, while of undisputed importance in numerical analysis, is notorious for the difficulties which it presents when it *is to be taught in the classroom in such a manner that the student is neither insulted by lack of mathematical content nor bored by lack of transparence and clarity.*

-PETER HENRICI, *A Model for the Propagation of Rounding Error in Floating Arithmetic* (1980)

The two main classes of rounding error analysis are not, as my audience might imagine, 'backwards' and 'forwards', but rather 'one's own' and 'other people's'. One's own is, of course, a model of lucidity; that of others serves only to obscure the essential simplicity of the matter in hand.

-J. H. WILKINSON, *The State of the Art in Error Analysis (1985)*

Having defined a model for floating point arithmetic in the last chapter, we now apply the model to some basic matrix computations, beginning with inner products. This first application is simple enough to permit a short analysis, yet rich enough to illustrate the ideas of forward and backward error. It also raises the thorny question of what is the best notation to use in an error analysis. We introduce the "γ_n" notation, which we use widely, though not exclusively, in the book. The inner product analysis leads immediately to results for matrix-vector and matrix-matrix multiplication.

In the last two sections we determine a model for rounding errors in complex arithmetic and derive some miscellaneous results of use in later chapters.

3.1. Inner and Outer Products

Consider the inner product $s_n = x^T y$, where $x, y \in \mathbb{R}^n$. Since the order of evaluation of $s_n = x_1y_1 + \dots + x_ny_n$ affects the analysis (but not the final error *bounds),* we will assume that the evaluation is from left to right. (The effect of particular orderings is discussed in detail in Chapter 4, which considers the special case of summation.) In the following analysis, and throughout the book, a hat denotes a computed quantity.

Let $s_i = x_1y_1 + \dots + x_iy_i$ denote the ith partial sum. Using the standard model (2.4), we have

$$\begin{aligned}\widehat{s}_1 &= fl(x_1y_1) = x_1y_1(1+\delta_1),\\ \widehat{s}_2 &= fl(\widehat{s}_1 + x_2y_2) = \big(\widehat{s}_1 + x_2y_2(1+\delta_2)\big)(1+\delta_3)\\ &= \big(x_1y_1(1+\delta_1) + x_2y_2(1+\delta_2)\big)(1+\delta_3)\\ &= x_1y_1(1+\delta_1)(1+\delta_3) + x_2y_2(1+\delta_2)(1+\delta_3),\end{aligned} \tag{3.1}$$

where $|\delta_i| \le u$, $i = 1{:}3$. For our purposes it is not necessary to distinguish between the different δ_i terms, so to simplify the expressions let us drop the subscripts on the δ_i and write $1 + \delta_i \equiv 1 \pm \delta$. Then

$$\begin{aligned}\widehat{s}_3 &= fl(\widehat{s}_2 + x_3y_3) = \big(\widehat{s}_2 + x_3y_3(1\pm\delta)\big)(1\pm\delta)\\ &= \big(x_1y_1(1\pm\delta)^2 + x_2y_2(1\pm\delta)^2 + x_3y_3(1\pm\delta)\big)(1\pm\delta)\\ &= x_1y_1(1\pm\delta)^3 + x_2y_2(1\pm\delta)^3 + x_3y_3(1\pm\delta)^2.\end{aligned}$$

The pattern is clear. Overall, we have

$$\widehat{s}_n = x_1y_1(1\pm\delta)^n + x_2y_2(1\pm\delta)^n + x_3y_3(1\pm\delta)^{n-1} + \dots + x_ny_n(1\pm\delta)^2. \tag{3.2}$$

There are various ways to simplify this expression. A particularly elegant way is to use the following result.

Lemma 3.1. *If* $|\delta_i| \le u$ *and* $\rho_i = \pm 1$ *for* $i = 1{:}n$, *and* $nu < 1$, *then*

$$\prod_{i=1}^{n}(1+\delta_i)^{\rho_i} = 1 + \theta_n,$$

where

$$|\theta_n| \le \frac{nu}{1-nu} =: \gamma_n.$$

Proof. See Problem 3.1. □

The θ_n, and γ_n notation will be used throughout this book. Whenever we write γ_n there is an implicit assumption that $nu < 1$, which is true in virtually any circumstance that might arise with IEEE single or double precision arithmetic.

Applying the lemma to (3.2) we obtain

$$\widehat{s}_n = x_1y_1(1+\theta_n) + x_2y_2(1+\theta'_n) + x_3y_3(1+\theta_{n-1}) + \cdots + x_ny_n(1+\theta_2). \quad (3.3)$$

This is a backward error result and may be interpreted as follows: the computed inner product is the exact one for a perturbed set of data $x_1, \dots, x_n$, $y_1(1+\theta_n), y_2(1+\theta'_n), \dots, y_n(1+\theta_2)$ (alternatively, we could perturb the x_i and leave the y_i alone). Each relative perturbation is certainly bounded by $\gamma_n = nu/(1-nu)$, so the perturbations are tiny.

The result (3.3) applies to one particular order of evaluation. It is easy to see that for any order of evaluation we have, using vector notation,

$$fl(x^Ty) = (x+\Delta x)^Ty = x^T(y+\Delta y), \quad |\Delta x| \le \gamma_n|x|, \quad |\Delta y| \le \gamma_n|y|, \quad (3.4)$$

where $|x|$ denotes the vector with elements $|x_i|$ and inequalities between vectors (and, later, matrices) hold componentwise.

A forward error bound follows immediately from (3.4):

$$|x^Ty - fl(x^Ty)| \le \gamma_n \sum_{i=1}^{n} |x_iy_i| = \gamma_n|x|^T|y|. \quad (3.5)$$

If $y = x$, so that we are forming a sum of squares x^Tx, this result shows that high relative accuracy is obtained. However, in general, high relative accuracy is not guaranteed if $|x^Ty| \ll |x|^T|y|$.

It is easy to see that precisely the same results (3.3)-(3.5) hold when we use the no-guard-digit rounding error model (2.6). For example, expression (3.1) becomes $\widehat{s}_2 = x_1y_1(1+\delta_1)(1+\delta_3) + x_2y_2(1+\delta_2)(1+\delta_4)$, where δ_4 has replaced a second occurrence of δ_3, but this has no effect on the error bounds.

It is worth emphasizing that the constants in the bounds above can be reduced by focusing on particular implementations of an inner product. For example, if $n = 2m$ and we compute

$$s_1 = x(1\colon m)^T y(1\colon m)$$
$$s_2 = x(m+1\colon n)^T y(m+1\colon n)$$
$$s_n = s_1 + s_2$$

then $|s_n - \widehat{s}_n| \le \gamma_{n/2+1}|x^T||y|$. By accumulating the inner product in two pieces we have almost halved the error bound. This idea can be generalized by breaking the inner product into k pieces, with each mini inner product of length n/k being evaluated separately and the results summed. The error bound is now $\gamma_{n/k+k-1}|x^T||y|$, which achieves its minimal value of $\gamma_{2\sqrt{n}-1}|x^T||y|$ for $k = \sqrt{n}$ (or, rather, we should take k to be the nearest integer to $\sqrt{n}$). But it is possible to do even better by using pairwise summation of the products x_iy_i (this method is described in §4.1). With pairwise summation, the error bound becomes

$$|s_n - \widehat{s}_n| \le \gamma_{\lceil \log_2 n\rceil+1}|x^T||y|.$$

Since many of the error analyses in this book are built upon the error analysis of inner products, it follows that the constants in these higher level bounds can also be reduced by using one of these nonstandard inner product implementations. The main significance of this observation is that we should not attach too much significance to the precise values of the constants in error bounds.

Inner products are amenable to being calculated in extended precision. If the working precision involves a t-digit mantissa then the product of two floating point numbers has a mantissa of $2t-1$ or $2t$ digits and so can be represented exactly with a $2t$-digit mantissa. Some computers always form the $2t$-digit product before rounding to t digits. thus allowing an inner product to be accumulated at $2t$-digit precision at little or no extra cost, prior to a final rounding.

The extended precision computation can be expressed as $fl(fl_e(x^Ty))$, where fl_e denotes computations with unit roundoff u_e ($u_e < u$). Defining $\widehat{s}_n = fl_e(x^Ty)$, the analysis above holds with u replaced by u_e in (3.3) (and with the subscripts on the θ_i, reduced by 1 if the multiplications are done exactly). For the final rounding we have

$$fl(fl_e(x^Ty)) = \widehat{s}_n(1+\delta), \qquad |\delta| \le u,$$

and so, overall,

$$|x^Ty - fl(fl_e(x^Ty))| \le u|x^Ty| + \frac{nu_e}{1-nu_e}(1+u)|x|^T|y|.$$

Hence, as long as $nu_e(|x|^T|y| \le u|x^Ty|$, the computed inner product is about as good as the rounded exact inner product. The effect of using extended precision inner products in an algorithm is typically to enable a factor n to be removed from the overall error bound.

Because extended precision inner product calculations are machine dependent it is difficult or impossible to write portable programs that use them. Most modern numerical codes (for example those in EISPACK, LINPACK, and LAPACK) do not use extended precision inner products. One process in which these more accurate products are needed is the traditional formulation of iterative refinement, in which the aim is to improve the accuracy of the computed solution to a linear system (see Chapter 11).

We have seen that computation of an inner product is a backward stable process. What can be said for an outer product $A = xy^T$, where $x, y, \in \mathbb{R}^n$? The analysis is easy. We have $\hat{a}_{ij} = x_i y_i(1 + \delta_{ij})$, $|\delta_{ij}| \le u$, so

$$\hat{A} = xy^T + \Delta, \qquad |\Delta| \le u|xy^T|. \tag{3.6}$$

This is a satisfying result, but the computation is not backward stable. In fact, $\hat{A} = (x + \Delta x)(y + \Delta y)^T$ does not hold for *any* Δx and Δy (let alone a small Δx and Δy) because A is not in general a rank 1 matrix.

This distinction between inner and outer products illustrates a general principle: a numerical process is more likely to be backward stable when the number of outputs is small compared with the number of inputs, so that there is an abundance of data onto which to “throw the backward error”. An inner product has the minimum number of outputs for its $2n$ scalar inputs, and an outer product has the maximum number (among standard linear algebra operations).

3.2. The Purpose of Rounding Error Analysis

Before embarking on further error analyses, it is worthwhile to consider what a rounding error analysis is designed to achieve. The purpose is to show the existence of an a priori bound for some appropriate measure of the effects of rounding errors on an algorithm. Whether a bound exists is the most important question. Ideally, the bound is small for all choices of problem data. If not, it should reveal features of the algorithm that characterize any potential instability, and thereby suggest how the instability can be cured or avoided. For some unstable algorithms, however, there is no useful error bound. (For example, no bound is known for the loss of orthogonality due to rounding error in the classical Gram-Schmidt method; see §18.7.)

The constant terms in an error bound (those depending only on the problem dimensions) are the least important parts of it. As discussed in §2.6, the constants usually cause the bound to overestimate the actual error by orders of magnitude. It is not worth spending much effort to minimize constants because the achievable improvements are usually insignificant.

It is worth spending effort, though, to put error bounds in a concise, easily interpreted form. Part of the secret is in the choice of notation, which we discuss in §3.4, including the question of what symbols to choose for variables (see the discussion in Higham [554, 1993, §3.5]).

If sharp error estimates or bounds are desired they should be computed a posteriori, so that the actual rounding errors that occur can be taken into account. One approach is to use running error analysis, described in the next section. Other possibilities are to compute the backward error explicitly, as can be done for linear equation and least squares problems (see §§7.1, 7.2, and 19.7), or to apply iterative refinement to obtain a correct ion that approximates the forward error (see Chapter 11).

3.3. Running Error Analysis

The forward error bound (3.5) is an a priori bound that does not depend on the actual rounding errors committed. We can derive a sharper, a posteriori bound by reworking the analysis. The inner product evaluation may be expressed as

```
$s_0 = 0$
for $i = 1:n$
    $s_i = s_{i-1} + x_i y_i$
end
```

Write the computed partial sums as $\widehat{s}_i =: s_i + e_i$ and let $\widehat{z}_i := fl(x_i y_i)$. We have, using (2.5),

$$\widehat{z}_i = \frac{x_i y_i}{1+\delta_i}, \qquad |\delta_i| \le u \qquad \Rightarrow \qquad \widehat{z}_i = x_i y_i - \delta_i \widehat{z}_i.$$

Similarly, $(1+\epsilon_i)\widehat{s}_i = \widehat{s}_{i-1} + \widehat{z}_i$, where $|\epsilon_i| \le u$, or

$$s_i + e_i + \epsilon_i \widehat{s}_i = s_{i-1} + e_{i-1} + x_i y_i - \delta_i \widehat{z}_i.$$

Hence $e_i = e_{i-1} - \epsilon_i \widehat{s}_i - \delta_i \widehat{z}_i$, which gives

$$|e_i| \le |e_{i-1}| + u|\widehat{s}_i| + u|\widehat{z}_i|.$$

Since $e_0 = 0$, we have $|e_n| \le u\mu_n$, where

$$\mu_i = \mu_{i-1} + |\widehat{s}_i| + |\widehat{z}_i|, \qquad \mu_0 = 0.$$

Algorithm 3.2. Given $x, y \in \mathbb{R}^n$ this algorithm computes $s = fl(x^T y)$ and a number μ such that $|s - x^T y| \le \mu$.

```
    $s = 0$
    $\mu = 0$
    for $i = 1:n$
        $z = x_i y_i$
        $s = s + z$
    $\mu = \mu + |s| + |z|$
  end
$\mu = \mu * \mu$
```

This type of computation, where an error bound is computed concurrently with the solution, is called *running error analysis*. The underlying idea is simple: we use the modified form (2.5) of the standard rounding error model to write

$$|x \text{ op } y - fl(x \text{ op } y)| \le u|fl(x \text{ op } y)|,$$

which gives a bound for the error in x op y that is easily computed, since $fl(x \text{ op } y)$ is stored on the computer. Key features of a running error analysis are that few inequalities are involved in the derivation of the bound and that the actual computed intermediate quantities are used, enabling advantage to be taken of cancellation and zero operands. A running error bound is, therefore, usually smaller than an a priori one.

There are, of course, rounding errors in the computation of the running error bound, but their effect is negligible for $nu \ll 1$ (we do not need many correct significant digits in an error bound).

Running error analysis is a somewhat neglected practice nowadays, but it was widely used by Wilkinson in the early years of computing. It is applicable to almost any numerical algorithm. Wilkinson explains [1101, 1986]

> When doing running error analysis on the ACE at no time did I write down these expressions. I merely took an existing program (without any error analysis) and modified it as follows. As each arithmetic operation was performed I added the absolute value of the computed result (or of the dividend) into the accumulating error bound.

For more on the derivation of running error bounds see Wilkinson [1100, 1985] or [1101, 1986]. A running error analysis for Horner's method is given in §5.1.

3.4. Notation for Error Analysis

Another way to write (3.5) is

$$|x^T y - fl(x^T y)| \le nu|x|^T|y| + O(u^2). \tag{3.7}$$

In general, which form of bound is preferable depends on the context. The use of first-order bounds such as (3.7) can simplify the algebra in an analysis. But there can be some doubt as to the size of the constant term concealed by the big-oh notation. Furthermore, in vector inequalities an $O(u^2)$ term hides the structure of the vector it is bounding and so can make interpretation of the result difficult; for example, the inequality $|x - y| \le nu|x| + O(u^2)$ does not imply that y approximates every element of x with the same relative error (indeed the relative error could be infinite when $x_i = 0$, as far as we can tell from the bound).

In more complicated analyses based on Lemma 3.1 it is necessary to manipulate the $1 + \theta_k$ and γ_k terms. The next, lemma provides the necessary rules.

Lemma 3.3. *For any positive integer k let θ_k denote a quantity bounded according to $|\theta_k| \le \gamma_k = ku/(1 - ku)$. The following relations hold:*

$$\begin{aligned}
(1+\theta_k)(1+\theta_j) &= 1+\theta_{k+j},\\
\frac{1+\theta_k}{1+\theta_j} &= \begin{cases} 1+\theta_{k+j}, & j \le k,\\ 1+\theta_{k+2j}, & j > k,\end{cases}\\
\gamma_k\gamma_j &\le \gamma_{\min(k,j)},\\
i\gamma_k &\le \gamma_{ik},\\
\gamma_k + u &\le \gamma_{k+1},\\
\gamma_k + \gamma_j + \gamma_k\gamma_j &\le \gamma_{k+j}.
\end{aligned}$$

Proof. See Problem 3.4. □

Concerning the second rule in Lemma 3.3, we certainly have

$$\prod_{i=1}^{k}(1+\delta_i)^{\pm 1} \Big/ \prod_{i=1}^{j}(1+\delta_i)^{\pm 1} = 1+\theta_{k+j},$$

but if we are given only the expression $(1 + \theta_k)/(1 + \theta_j)$ and the bounds for θ_k and θ_j, we cannot do better than to replace it by θ_{k+2j} for $j > k$.

Another style of writing bounds is made possible by the following lemma.

Lemma 3.4. *If $|\delta| \le u$ for $i = 1:n$ and $nu < 0.01$, then*

$$\prod_{i=1}^{n}(1+\delta_i) = 1+\eta_n,$$

where $|\eta_n| \le 1.01nu$.

Proof. We have

$$|\eta_n| = \left|\prod_{i=1}^{n}(1+\delta_i) - 1\right| \le (1+u)^n - 1.$$

Since $1 + x \le e^x$ for $x \ge 0$, we have $(1 + u)^n < \exp(nu)$, and so

$$\begin{aligned}(1+u)^n - 1 &< nu + \frac{(nu)^2}{2!} + \frac{(nu)^3}{3!} + \cdots \\ &< nu\left(1 + \frac{nu}{2} + \left(\frac{nu}{2}\right)^2 + \left(\frac{nu}{2}\right)^3 + \cdots\right) \\ &= nu\frac{1}{1-nu/2} \\ &< nu\frac{1}{0.995} < 1.01nu. \qquad \square\end{aligned}$$

Note that this lemma is slightly stronger than the corresponding bound we can obtain from Lemma 3.1: $|\theta_n| \le nu/(1 - nu) < nu/0.99 = 1.0101\ldots nu$.

Lemma 3.4 enables us to derive, as an alternative to (3.5),

$$|x^T y - fl(x^T y)| \le 1.01nu|x|^T|y|. \tag{3.8}$$

A convenient device for keeping track of powers of $1 + \delta$ terms was introduced by Stewart [941, 1973, App. 3]. His relative error counter $<k>$ denotes a product

$$<k> = \prod_{i=1}^{k}(1+\delta_i)^{\rho_i}, \qquad \rho_i = \pm 1, \quad |\delta_i| \le u. \tag{3.9}$$

The counters can be manipulated using the rules

$$\begin{aligned} <j><k> &= <j+k>, \\ \frac{<j>}{<k>} &= <j+k>. \end{aligned}$$

At the end of an analysis it is necessary to bound $|<k> - 1|$, for which any of the techniques described above can be used.

Wilkinson explained in [1100, 1985] that he used a similar notation in his research, writing ψ^r for a product of r factors $1 + \delta_i$ with $|\delta_i| \le u$. He also derived results for specific values of n, before treating the general case-a useful trick of the trade!

An alternative notation to $fl(\cdot)$ to denote the rounded value of a number or the computed value of an expression is $[\cdot]$, suggested by Kahan. Thus, we would write $[a + [b * c]]$ instead of $fl(a + fl(b * c))$.

A completely different notation for error analysis has been proposed by Olver [807, 1978], and subsequently used by him and several other authors. For scalars x and y of the same sign, Olver defines the *relative precision* rp as follows:

$$y \approx x;\ \mathrm{rp}(a) \text{ means that } y = e^{\delta} x, \quad |\delta| \le a.$$

Since $e^{\delta} = 1 + \delta + O(\delta^2)$, this definition implies that the relative error in x as an approximation to y (or vice versa) is at most $a + O(a^2)$. But, unlike the usual definition of relative error, the relative precision possesses the properties of

$$\text{symmetry:}\quad y \approx x;\ \mathrm{rp}(a) \iff x \approx y;\ \mathrm{rp}(a),$$
$$\text{additivity:}\quad y \approx x;\ \mathrm{rp}(a) \text{ and } z \approx y;\ \mathrm{rp}(\beta) \implies z \approx x\text{: } \mathrm{rp}(a + \beta).$$

Proponents of relative precision claim that the symmetry and additivity properties make it easier to work with than the relative error.

Pryce [845, 1981] gives an excellent appraisal of relative precision, with many examples. He uses the additional notation $1(\delta)$ to mean a number θ with $\theta \approx 1;\ \mathrm{rp}(\delta)$. The $1(\delta)$ notation is the analogue for relative precision of Stewart's $<k>$ counter for relative error. In later papers, Pryce extends the rp notation to vector and matrices and shows how it can be used in the error analysis of some basic matrix computations [846, 1984], [847, 1985].

Relative precision has not achieved wide use. The important thing for an error analyst is to settle on a comfortable notation that does not hinder the thinking process. It does not really matter which of the notations described in this section is used, as long as the final result is informative and expressed in a readable form.

3.5. Matrix Multiplication

Given error analysis for inner products it is straightforward to analyse matrix-vector and matrix-matrix products. Let $A \in \mathbb{R}^{m\times n}$, $x \in \mathbb{R}^n$ and $y = Ax$. The vector y can be formed as m inner products, $y_i == a_i^T x$, $i = 1:m$, where a_i^T is the ith row of A. From (3.4) we have

$$\widehat{y}_i = (a_i + \Delta a_i)^T x, \qquad |\Delta a_i| \le \gamma_n |a_i|.$$

This gives the backward error result

$$\widehat{y} = (A + \Delta A)x, \qquad |\Delta A| \le \gamma_n |A| \qquad (A \in \mathbb{R}^{m\times n}), \tag{3.10}$$

which implies the forward error bound

$$|y - \widehat{y}| \le \gamma_n |A||x|. \tag{3.11}$$

Normwise bounds readily follow (see Chapter 6 for norm definitions): for example,

$$\|y-\widehat{y}\|_p \le \gamma_n \|A\|_p \|x\|_p, \qquad p = 1, \infty.$$

This inner product formation of y can be expressed algorithmically as

```
% Sdot or inner product form.
$y(1{:}m) = 0$
for $i = 1{:}m$
    for $j = 1{:}n$
        $y(i) = y(i) + a(i,j)\,x(j)$
    end
end
```

The two loops can be interchanged to give

```
% Saxpy form.
$y(1{:}m) = 0$
for $j = 1{:}n$
    for $i = 1{:}m$
        $y(i) = y(i) + a(i,j)\,x(j)$
    end
end
```

The terms "sdot" and "saxpy" come from the BLAS (see §D.1). Sdot stands for (single precision) dot product, and saxpy for (single precision) a times x plus y. The question of interest is whether (3.10) and (3.11) hold for the saxpy form. They do: the saxpy algorithm still forms the inner products $a_i^T x$, but instead of forming each one in turn it evaluates them all "in parallel", a term at a time. The key observation is that exactly the same operations are performed, and hence *exactly the same rounding errors are committed-the* only difference is the order in which the rounding errors are created.

This "rounding error equivalence" of algorithms that are mathematically identical but algorithmically different is one that occurs frequently in matrix computations. The equivalence is not always as easy to see as it is for matrix-vector products.

Now consider matrix multiplication: $C = AB$, where $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$. Matrix multiplication is a triple loop procedure, with six possible loop orderings, one of which is

```
$C(1{:}m, 1{:}p) = 0$
for $i = 1{:}m$
    for $j = 1{:}p$
        for $k = 1{:}n$
            $C(i,j) = C(i,j) + A(i,k)\,B(k,j)$
        end
    end
end
```

As for the matrix -vector product. all six versions commit the same rounding errors, so it suffices to consider any one of them. The “jik” and “jki” orderings both compute C a column at a time: $c_j = Ab_j$, where $c_j = C(:,j)$ and $b_j = B(:,j)$. From (3.10),

$$\widehat{c}_j = (A + \Delta A_j)b_j, \qquad |\Delta A_j| \le \gamma_n |A|.$$

Hence the jth computed column of C has a small backward error: it is the exact jth column for slightly perturbed data. The same cannot be said for $\widehat{C}$ as a whole (see Problem 3.5 for a possibly large backward error bound). However, we have the forward error bound

$$|C - \widehat{C}| \le \gamma_n |A||B| \qquad (A \in \mathbb{R}^{m\times n},\ B \in \mathbb{R}^{n\times p}), \tag{3.12}$$

and the corresponding normwise bounds include

$$\|C - \widehat{C}\|_p \le \gamma_n \|A\|_p \|B\|_p, \qquad p = 1, 2, F.$$

The bound (3.12) falls short of the ideal bound $|C - \widehat{C}| \le \gamma_n |C|$, which says that each component of C is computed with high relative accuracy. Nevertheless (3.12) is the best bound we can expect, because it reflects the sensitivity of the product to componentwise relative perturbations in the data: for any i and j we can find a perturbation ΔA with $|\Delta A| \le u|A|$ such that $|(A+\Delta A)B-AB|_{ij}=u(|A||B|)_{ij}$ (similarly for perturbations in B).

3.6. Complex Arithmetic

To carry out error analysis of algorithms in complex arithmetic we need a model for the basic arithmetic operations. Since complex arithmetic must be implemented using real arithmetic, the complex model is a consequence of the corresponding real one. We will assume that for complex numbers $x = a + ib$ and $y = c + id$ we compute

$$x \pm y = a + c \pm i(b + d), \tag{3.13a}$$

$$xy = ac - bd + i(ad + bc), \tag{3.13b}$$

$$x/y = \frac{ac + bd}{c^2 + d^2} + i\frac{bc - ad}{c^2 + d^2}. \tag{3.13c}$$

Lemma 3.5. *For $x, y \in \mathbb{C}$ the basic arithmetic operations computed according to* (3.13) *under the standard model* (2.4) *satisfy*

$$\begin{aligned} fl(x \pm y) &= (x+y)(1+\delta), && |\delta| \le u,\\ fl(xy) &= xy(1+\delta), && |\delta| \le \sqrt{2}\gamma_2,\\ fl(x/y) &= (x/y)(1+\delta), && |\delta| \le \sqrt{2}\gamma_4. \end{aligned}$$

Proof. Throughout the proof, δ_i denotes a number bounded by $|\delta_i| \le u$. Addition/subtraction:

$$\begin{aligned} fl(x+y) &= (a+c)(1+\delta_1) + i(b+d)(1+\delta_2)\\ &= x + y + (a+c)\delta_1 + i(b+d)\delta_2, \end{aligned}$$

so

$$|fl(x+y) - (x+y)|^2 \le (|a+c|^2 + |b+d|^2)u^2 = (|x+y|u)^2,$$

as required.

Multiplication:

$$\begin{aligned} fl(xy) &= \big(ac(1+\delta_1) - bd(1+\delta_2)\big)(1+\delta_3)\\ &\qquad + i\big(ad(1+\delta_4) + bc(1+\delta_5)\big)(1+\delta_6)\\ &= ac(1+\theta_2) - bd(1+\theta_2') + i\big(ad(1+\theta_2'') + bc(1+\theta_2''')\big)\\ &= xy + e, \end{aligned}$$

where

$$\begin{aligned} |e|^2 &\le \gamma_2^2\big((|ac|+|bd|)^2 + (|ad|+|bc|)^2\big)\\ &\le 2\gamma_2^2(a^2+b^2)(c^2+d^2)\\ &= 2\gamma_2^2|xy|^2, \end{aligned}$$

as required.

Division:

$$\begin{aligned} fl(c^2+d^2) &= \big(c^2(1+\delta_1) + d^2(1+\delta_2)\big)(1+\delta_3)\\ &= c^2(1+\theta_2) + d^2(1+\theta_2')\\ &= (c^2+d^2)(1+\theta_2''). \end{aligned}$$

Then

$$\begin{aligned} fl(\mathrm{Re}\, x/y) &= \frac{\big(ac(1+\delta_4) + bd(1+\delta_5)\big)(1+\delta_6)}{(c^2+d^2)(1+\theta_2'')}\\ &= \frac{ac(1+\theta_2''') + bd(1+\theta_2'''')}{(c^2+d^2)(1+\theta_2'')}\\ &= \mathrm{Re}\, x/y + e_1, \end{aligned}$$

where, using Lemma 3.3,

$$|e_1| \le \frac{|ac| + |bd|}{c^2 + d^2}\gamma_4.$$

Using the analogous formula for the error in *fl*(Imx/y),

$$\begin{aligned} - x/y|^2 &\le \frac{(|ac| + |bd|)^2 + (|bc| + |ad|)^2}{(c^2 + d^2)^2}\gamma_4^2 \\ &\le \frac{2(a^2 + b^2)(c^2 + d^2)}{(c^2 + d^2)^2}\gamma_4^2 \\ &= 2\gamma_4^2|x/y|^2, \end{aligned}$$

which completes the proof. □

It is worth stressing that δ in Lemma 3.5 is a complex number, so we cannot conclude from the lemma that the real and imaginary parts of *fl*(x op y) are obtained to high relative accuracy---only that they are obtained to high accuracy relative to $|x$ op $y|$.

As explained in §25.8, the formula (3.13c) is not recommended for practical use since it is susceptible to overflow. For the alternative formula (25.1), which avoids overflow, similar analysis to that in the proof of Lemma 3.5 shows that

$$fl(x/y) = (x/y)(1 + \delta), \qquad |\delta| \le \sqrt{2}\gamma_7.$$

Bounds for the rounding errors in the basic complex arithmetic operations are rarely given in the literature. Indeed, virtually all published error analyses in matrix computations are for real arithmetic. However, because the bounds of Lemma 3.5 are of the same form as for the standard model (2.4) for real arithmetic, most results for real arithmetic (including virtually all those in this book) are valid for complex arithmetic, provided that the constants are increased appropriately.

3.7. Miscellany

In this section we give some miscellaneous results that will be needed in later chapters. The first two results provide convenient ways to bound the effect of perturbations in a matrix product. The first result uses norms and the second. components.

Lemma 3.6. *If* $X_j + \Delta X_j \in$ IR$^{n \times n}$ *satisfies* $||DX_j|| \le d_j||X_j||$ *for all* j *for a consistent norm, then*

$$\left\| \prod_{j=0}^{m}(X_j + \Delta X_j) - \prod_{j=0}^{m} X_j \right\| \le \left(\prod_{j=0}^{m}(1 + \delta_j) - 1 \right) \prod_{j=0}^{m} \|X_j\|.$$

Proof. The proof is a straightforward induction, which we leave as an exercise (Problem 3.10). □

A componentwise result is entirely analogous.

Lemma 3.7. *If* $X_j + \Delta X_j \in \mathbb{R}^{n\times n}$ *satisfies* $|\Delta X_j| \le \delta_j |X_j|$ *for all* j, *then*

$$\left|\prod_{j=0}^{m}(X_j + \Delta X_j) - \prod_{j=0}^{m} X_j\right| \le \left(\prod_{j=0}^{m}(1+\delta_j) - 1\right)\prod_{j=0}^{m}|X_j|. \qquad \square$$

The final result describes the computation of the "rank 1 update" $y = (I - ab^T)x$, which is an operation arising in various algorithms, including the Gram-Schmidt method and Householder QR factorization.

Lemma 3.8. *Let* $a, b, x \in \mathbb{R}^n$ *and let* $y = (I - ab^T)x$ *be computed as* $\widehat{y} = fl(x - a(b^Tx))$. *Then* $\widehat{y} = y + \Delta y$, *where*

$$|\Delta y| \le \gamma_{n+3}(I + |a||b^T|)|x|,$$

so that

$$\|\Delta y\|_2 \le \gamma_{n+3}(1 + \|a\|_2\|b\|_2)\|x\|_2.$$

Proof. Consider first the computation of $w = a(b^Tx)$. We have

$$\begin{aligned}\widehat{w} &:= (a+\Delta a)b^T(x+\Delta x), \qquad |\Delta a| \le u|a|, \quad |\Delta x| \le \gamma_n|x|,\\ &= a(b^Tx) + a(b^T\Delta x) + \Delta a b^T(x+\Delta x)\\ &=: w + \Delta w,\end{aligned}$$

where

$$|\Delta w| \le \big(\gamma_n + u(1+\gamma_n)\big)|a||b^T||x|.$$

Finally, $\widehat{y} = fl(x - \widehat{w})$ satisfies

$$\widehat{y} = x - a(b^Tx) - \Delta w + \Delta y_1, \qquad |\Delta y_1| \le u(|x| + |\widehat{w}|),$$

and

$$|-\Delta w + \Delta y_1| \le u(|x| + |a||b^T||x|) + (1+u)\big(\gamma_n + u(1+\gamma_n)\big)|a||b^T||x|.$$

Hence $\widehat{y} = y + \Delta y$, where

$$\begin{aligned}|\Delta y| &\le \big[uI + (2u + u^2 + \gamma_n + 2u\gamma_n + u^2\gamma_n)|a||b^T|\big]|x|\\ &\le \gamma_{n+3}(I + |a||b^T|)|x|. \qquad \square\end{aligned}$$

3.8. Error Analysis Demystified

The principles underlying an error analysis can easily be obscured by the details. It is therefore instructive to examine the basic mechanism of forward and backward error analyses. We outline a general framework that reveals the essential simplicity.

Consider the problem of computing $z = f(a)$, where $f : \mathrm{IR}^n \longrightarrow \mathrm{IR}^m$. Any algorithm for computing z can be expressed as follows. Let $x_1 = a$ and $x_{k+1} = g_k(x_k)$, $k = 1{:}p$, where

$$x_{k+1} = \begin{bmatrix} x_k \\ \xi_k \end{bmatrix}, \qquad \xi_k \in \mathbb{R}.$$

The kth stage of the algorithm represents a single floating point operation and x_k contains the original data together with all the intermediate quantities computed so far. Finally, $z = \overline{I}x_{p+1}$, where $\overline{I}$ is comprised of a subset of the columns of the identity matrix (so that each z_i is a component of x_{p+1}). In floating point arithmetic we have

$$\widehat{x}_{k+1} = g_k(\widehat{x}_k) + \Delta x_{k+1},$$

where Δ_{xk+i} represents the rounding errors on the kth stage and should be easy to bound. We assume that the functions g_k are continuously differentiable and denote the Jacobian of g_k at a by J_k. Then, to first order,

$$\begin{aligned}
\widehat{x}_2 &= g_1(a) + \Delta x_2, \\
\widehat{x}_3 &= g_2(\widehat{x}_2) + \Delta x_3 = g_2(g_1(a) + \Delta x_2) + \Delta x_3 \\
&= g_2(g_1(a)) + J_2 \Delta x_2 + \Delta x_3, \\
\widehat{x}_4 &= g_3(\widehat{x}_3) + \Delta x_4 = g_3\big(g_2(g_1(a)) + J_2\Delta x_2 + \Delta x_3\big) + \Delta x_4 \\
&= g_3\big(g_2(g_1(a))\big) + J_3 J_2 \Delta x_2 + J_3 \Delta x_3 + \Delta x_4.
\end{aligned}$$

The pattern is clear: for the final $\widehat{z} = \widehat{x}_{p+1}$ we have

$$\begin{aligned}
\widehat{z} &= \overline{I}\big[g_p(\dots g_2(g_1(a))\dots) + J_p \dots J_2 \Delta x_2 + J_p \dots J_3 \Delta x_3 + \cdots \\
&\qquad\qquad + J_p \Delta x_p + \Delta x_{p+1}\big] \\
&= f(a) + \overline{I}\,\mathrm{diag}(J_p \dots J_2, \dots, J_p, I) \begin{bmatrix} \Delta x_2 \\ \Delta x_3 \\ \vdots \\ \Delta x_{p+1} \end{bmatrix} =: f(a) + Jh.
\end{aligned}$$

In a forward error analysis we bound $f(a) - \widehat{z}$, which requires bounds for (products of) the Jacobians J_k. In a backward error analysis we write, again to first order,

$$f(a) + Jh = \widehat{z} = f(a + \Delta a) = f(a) + J_f \Delta a,$$

where J_f is the Jacobian of f. So we need to solve, for Δa,

$$\underbrace{J_f}_{m\times n}\underbrace{\Delta a}_{n\times 1} = \underbrace{J}_{m\times q}\underbrace{h}_{q\times 1}, \qquad q = p(n+(p+1)/2).$$

In most matrix problems there are fewer outputs than inputs ($m < n$), so this is an underdetermined system. For a normwise backward error analysis we want a solution of minimum norm. For a componentwise backward error analysis, in which we may want (for example) to minimize ϵ subject to $|\Delta a| \le \epsilon |a|$, we can write

$$Jh = J_f D \cdot D^{-1}\Delta a =: Bc, \qquad D = \mathrm{diag}(a_i),$$

and then we want the solution c of minimal m-norm.

The conclusions are that forward error analysis corresponds to bounding derivatives and that backward error analysis corresponds to solving a large underdetermined linear system for a solution of minimal norm. In principal, therefore, error analysis is straightforward! Complicating factors in practice are that the Jacobians J_k may be difficult to obtain explicitly, that an error bound has to be expressed in a form that can easily be interpreted, and that we may want to keep track of higher-order terms.

3.9. Other Approaches

In this book we do not describe all possible approaches to error analysis. Some others are mentioned in this section.

Linearized rounding error bounds can be developed by applying equations that describe, to first order, the propagation of absolute or relative errors in the elementary operations +,-,*,/. The basics of this approach are given in many textbooks (see, for example, Dahlquist and Björck [262, 1974, §2.2] or Stoer and Bulirsch [955, 1980, §1.3]), but for a thorough treatment see Stummel [963, 1980], [964, 1981]. Ziv [1133, 1995] shows that linearized bounds can be turned into true bounds by increasing them by a factor that depends on the algorithm.

Rounding error analysis can be phrased in terms of graphs. This appears to have been first suggested by McCracken and Dorn [743, 1964], who use "process graphs" to represent a numerical computation and thereby to analyse the propagation of rounding errors. Subsequent more detailed treatments include those of Bauer [82, 1974], Miller [758, 1976], and Yalamov [1117, 1994]. The work on graphs falls under the heading of automatic error analysis (for more on which see Chapter 24) because processing of the large graphs required to represent practical computations is impractical by hand. Linnainmaa [705, 1976] shows how to compute the Taylor series expansion of the forward error in an algorithm in terms of the individual rounding errors, and he presents a graph framework for the computation.

Some authors have taken a computational complexity approach to error analysis, by aiming to minimize the number of rounding error terms in a forward error bound, perhaps by rearranging a computation. Because this approach ignores the possibility of cancellation of rounding errors, the results need to be interpreted with care. See Aggarwal and Burgmeier [6, 1979] and Tsao [1024, 1983].

3.10. Notes and References

The use of Lemma 3.1 for rounding error analysis appears to originate with the original 1972 German edition of a book by Stoer and Bulirsch [955, 1980]. The lemma is also used, with $p_i = 1$, by Shampine and Allen [911, 1973, p. 18].

Lemma 3.4 is given by Forsythe and Moler [396, 1967, p. 92]. Wilkinson made frequent use of a slightly different version of Lemma 3.4 in which the assumption is $nu < 0.1$ and the bound for $|h_n|$ is $1.06nu$ (see, e.g., [1089, 1965, p. 113]).

A straight forward notation for rounding errors that is subsumed by the notation described in this chapter is suggested by Scherer and Zeller [899, 1980].

Ziv [1131, 1982] proposes the relative error measure

$$d(x,y) = \|x-y\|/\max(\|x\|,\|y\|)$$

for vectors x and y and explains some of its favourable properties for error analysis.

Wilkinson [1089, 1965, p. 447] gives error bounds for complex arithmetic; Olver [808, 1983] does the same in the relative precision framework. Demmel [280, 1984] gives error bounds that extend those in Lemma 3.5 by taking into account the possibility of underflow.

Henrici [521, 1980] gives a brief, easy to read introduction to the use of the model (2.4) for analysing the propagation of rounding errors in a general algorithm. He uses a set notation that is another possible notation to add to those in §3.4.

The perspective on error analysis in §3.8 was suggested by J. W. Demmel.

Problems

3.1. Prove Lemma 3.1.

3.2. (Kielbasiniski and Schwetlick [658, 1988], [659, 1992]) Show that if $p_i \equiv 1$ in Lemma 3.1 then the stronger bound $|\phi_n| \le nu/(1-\tfrac{1}{2}nu)$ holds for $nu < 2$.

3.3. One algorithm for evaluating a continued fraction

$$a_0 + \cfrac{b_0}{a_1 + \cfrac{b_1}{a_2 + \cdots + \cfrac{b_n}{a_{n+1}}}}$$

is

```
$q_{n+1} = a_{n+1}$
for $k = n$:-1:0
    $q_k = a_k + b_k/q_{k+1}$
end
```

Derive a running error bound for this algorithm.

3.4. Prove Lemma 3.3.

3.5. (Backward error result for matrix multiplication.) Let $A \in \mathbb{R}^{n\times n}$ and $B \in \mathbb{R}^{n\times n}$ both be nonsingular. Show that $fl(AB) = (A + \Delta A)B$, where $|\Delta A| \le \gamma_n |A||B||B^{-1}|$, and derive a corresponding bound in which B is perturbed.

3.6. (Backward error definition for matrix multiplication.) Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ be of full rank and suppose $C \approx AB$. Define the componentwise backward error

$$\omega = \min\{\, \epsilon : C = (A + \Delta A)(B + \Delta B), \quad |\Delta A| \le \epsilon E,\ |\Delta B| \le \epsilon F \,\},$$

where E and F have nonnegative entries. Show that

$$\omega \ge \max_{i,j} \left(\sqrt{1 + \frac{|r_{ij}|}{g_{ij}}} - 1 \right),$$

where $R = C - AB$ and $G = EF$. Explain why the definition of w makes sense only when A and B have full rank. Define a mixed backward/forward error applicable to the general case.

3.7. Give analogues of the backward error results (3.4) and (3.10) for complex x, y, and A.

3.8. Let $A_1, \dots, A_k \in \mathbb{R}^{n\times n}$. Show that

$$\|A_1 \dots A_k - fl(A_1 \dots A_k)\|_F \le \bigl(kn^2 u + O(u^2)\bigr)\|A_1\|_2 \dots \|A_k\|_2.$$

3.9. Which is the more accurate way to compute $x^2 - y^2$: as $x^2 - y^2$ or as $(x+y)(x-y)$?(Assume the use of a guard digit. Note that this computation arises when squaring a complex number.)

3.10. Prove Lemma 3.6.

3.11. (Kahan [629, 1980]) Consider this MATLAB function, which returns the absolute value of its first argument $x \in \mathrm{IR}^n$:

```
function z = absolute(x , m )
y = x.-2;
for i=l:m
    y = sqrt(y);
end
z = y;
for i=l:m-1
    z = z.  -2;
end
```

Here is some output from a 486DX workstation that uses IEEE standard double precision arithmetic:

```
>> x = [.25 .5 .75 1.25 1.5 2]; z = absolute(x, 50); [x; z]
ans =
    0.2500    0.5000    0.7500    1.2500    1.5000    2.0000
    0.2528    0.5028    0.7788    1.2840    1.4550    2.1170
```

Give an error analysis to explain the results.

The same machine produced this output:

```
>> x = [.25 .5 .75 1.25 1.5 2]; z = absolute(x, 75); [x; z]
ans =
    0.2500    0.5000    0.7500    1.2500  1.5000    2.0000
    1.0000    1.0000    1.0000    1.0000  1.0000    1.0000
```

But a Sun SPARCstation, which also uses IEEE standard double precision arithmetic, produced

```
ans =
    0.2500    0.5000    0.7500    1.2500  1.5000    2.0000
         0         0         0    1.0000  1.0000    1.0000
```

Explain these results and why they differ (cf. §1.12.2).

3.12. Consider the quadrature rule

$$I(f) := \int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i) =: J(f),$$

where the weights w_i and nodes x_i are assumed to be floating point numbers. Assuming that the sum is evaluated in left-to-right order and that

$$fl(f(x_i)) = f(x_i)(1+\eta_i), \qquad |\eta_i| \le \eta,$$

obtain and interpret a bound for $|I(f) - \widehat{J}(f)|$, where $\widehat{J}(f) = fl(J(f))$.

Chapter 4
Summation

I do hate sums.
There is no greater mistake than to call arithmetic an exact *science.*
There are . . . hidden laws of Number
which it requires a mind like mine to perceive.
For instance, if you add a sum from the bottom up,
and then again from the top down,
the result is always different.

-MRS. LA TOUCHE[10]

Joseph Fourier introduced this delimited z-notation in 1820,
and it soon took the mathematical world by storm.

-RONALD L. GRAHAM, DONALD E. KNUTH, and
OREN PATASHNIK, *Concrete Mathematics (1989)*

One of the major difficulties in a practical [error] analysis
is that of description.
An ounce of analysis follows a pound of preparation.

-BERESFORD N. PARLETT, *Matrix Eigenvalue Problems* (1965)

---

[10] Quoted in Mathematical Gazette [730, 1924].

Sums of floating point numbers are ubiquitous in scientific computing. They occur when evaluating inner products, means, variances, norms, and all kinds of nonlinear functions. Although at first sight summation might appear to offer little scope for algorithmic ingenuity, the usual "recursive summation" (with various orderings) is just one of a variety of possible techniques. We describe several summation methods and their error analyses in this chapter. No one method is uniformly more accurate than the others, but some guidelines can be given on the choice of method in particular cases.

4.1. Summation Methods

In most circumstances in scientific computing we would naturally translate a sum $\sum_{i=1}^{n} x_i$ into code of the form

```
$s = 0$
for $i = 1:n$
    $s = s + x_i$
end
```

This is known *as recursive summation.* Since the individual rounding errors depend on the operands being summed, the accuracy of the computed sum $\widehat{s}$ varies with the ordering of the x_i. (Hence Mrs. La Touche, quoted at the beginning of the chapter, was correct if we interpret her remarks as applying to floating point arithmetic.) Two interesting choices of ordering are the increasing order $|x_1| \le |x_2| \le \dots \le |x_n|$, and the decreasing order $|x_1| \ge |x_2| \ge \dots \ge |x_n|$.

Another method is *pairwise summation* (also known as cascade summation, or fan-in summation), in which the x_i are summed in pairs according to

$$y_i = x_{2i-1} + x_{2i}, \quad i = 1 : \lfloor \tfrac{n}{2} \rfloor \quad (y_{(n+1)/2} = x_n \text{ if } n \text{ is odd}),$$

and this pairwise summation process is repeated recursively on the y_i, $i = 1:[(n+1)/2]$. The sum is obtained in $[\log_2 n]$ stages. For $n = 6$, for example, pairwise summation forms

$$S_6 = ((x_1 + x_2) + (x_3 + x_4)) + (x_5 + x_6).$$

Pairwise summation is attractive for parallel computing, because each of the $[\log_2 n]$ stages can be done in parallel [573, 1988, §5.2.2].

A third summation method is the *insertion* method. First, the x_i are sorted by order of increasing magnitude (alternatively, some other ordering could be used). Then $x_1 + x_2$ is formed, and the sum is inserted into the list $x_2, \dots, x_n$, maintaining the increasing order. The process is repeated recursively until the final sum is obtained. In particular cases the insertion method reduces to one of the other two. For example, if $x_i = 2^{i-1}$, the insertion method is equivalent to recursive summation, since the insertion is always to the bottom of the list:

$$1248 \quad \rightarrow \quad \underline{3}48 \quad \rightarrow \quad \underline{7}8 \quad \rightarrow \quad 15.$$

On the other hand, if $1 \le x_1 < x_2 < \dots < x_n \le 2$, every insertion is to the end of the list, and the method is equivalent to pairwise summation if n is a power of 2; for example, if $0 < \epsilon < 1/2$,

$$1,\ 1+\epsilon,\ 1+2\epsilon,\ 1+3\epsilon \quad \rightarrow \quad 1+2\epsilon,\ 1+3\epsilon,\ \underline{2+\epsilon} \quad \rightarrow \quad 2+\epsilon,\ \underline{2+5\epsilon} \quad \rightarrow \quad 4+6\epsilon.$$

To choose between these methods we need error analysis, which we develop in the next section.

4.2. Error Analysis

Error analysis can be done individually for the recursive, pairwise and insertion summation methods, but it is more profitable to recognize that each is a special case of a general algorithm and to analyse that algorithm.

Algorithm 4.1. Given numbers $x_1, \dots, x_n$ this algorithm computes $S_n = \sum_{i=1}^n x_i$.

```
Let $S = \{x_1, \dots, x_n\}$.
repeat while $S$ contains more than one element
        Remove two numbers $x$ and $y$ from $S$,
        and add their sum $x + y$ to $S$.
end
Assign the remaining element of $S$ to $S_n$.
```

Note that since there are n numbers to be added and hence $n - 1$ additions to be performed, there must be precisely $n - 1$ executions of the repeat loop.

First, let us check that the previous methods are special cases of Algorithm 4.1. Recursive summation (with any desired ordering) is obtained by taking x at each stage to be the sum computed on the previous stage of the algorithm. Pairwise summation is obtained by $[\log_2 n]$ groups of executions of the repeat loop, in each group of which the members of S are broken into pairs, each of which is summed. Finally, the insertion method is, by definition, a special case of Algorithm 4.1.

Now for the error analysis. Express the ith execution of the repeat loop as $T_i = x_{i_1} + y_{i_1}$. The computed sums satisfy (using (2.5))

$$\widehat{T}_i = \frac{x_{i_1} + y_{i_1}}{1 + \delta_i}, \quad |\delta_i| \le u, \quad i = 1{:}n-1. \tag{4.1}$$

The local error introduced in forming $\widehat{T}_i$ is $\delta_i\widehat{T}_i$. The overall error is the sum of the local errors (since summation is a linear process), so overall we have

$$E_n := S_n - \widehat{S}_n = \sum_{i=1}^{n-1} \delta_i\widehat{T}_i. \tag{4.2}$$

The smallest possible error bound is therefore

$$|E_n| \le u\sum_{i=1}^{n-1} |\widehat{T}_i|. \tag{4.3}$$

(This is actually in the form of a running error bound, because it contains the computed quantities-see §3.3.) It is easy to see that $|\widehat{T}_i| \le \sum_{j=1}^{n} |x_j| + O(u)$ for each i, and so we have also the weaker bound

$$|E_n| \le (n-1)u\sum_{i=1}^{n} |x_i| + O(u^2). \tag{4.4}$$

This is a forward error bound. A backward error result showing that $\widehat{S}_n$ is the exact sum of terms $x_i(1+\epsilon_i)$ with $|\epsilon_i| \le \gamma_{n-1}$ can be deduced from (4.1), using the fact that no number x_i takes part in more than n - 1 additions.

The following criterion is apparent from (4.2) and (-1.3):

> In designing or choosing a summation met hod to achieve high accuracy, the aim should be to minimize the absolute values of the intermediate sums T_i.

The aim specified in this criterion is surprisingly simple to state. When we consider specific methods, however, we find that the aim is difficult to achieve.

Consider recursive summation, for which $T_{i-1} = S_i := \sum_{j=1}^{i} x_j$. Ideally, we would like to choose the ordering of the x_i to minimize $\sum_{i=2}^{n} |\widehat{S}_i|$. This is a combinatorial optimization problem that is too expensive to solve in the context of summation. A reasonable compromise is to determine the ordering sequentially by minimizing, in turn, $|x_1|$, $|S_2|$, . . . , $|S_{n-1}|$. This ordering strategy, which we denote by Psum, can be implemented with $O(n \log n)$ comparisons. If we are willing to give up the property that the ordering is influenced by the signs of the x_i we can instead use the increasing ordering, which in general will lead to a larger value of $\sum_{i=2}^{n} |\widehat{S}_i|$ than that for the Psum ordering. If all the x_i have the same sign then all these orderings are equivalent. Therefore *when summing nonnegative numbers by recursive summation the increasing ordering is the best ordering, in the sense of having the smallest a priori forward error bound.*

How does the decreasing ordering fit into the picture? For the summation of positive numbers this ordering has little to recommend it. The bound (4.3) is no smaller, and potentially much larger, than it is for the increasing ordering. Furthermore, in a sum of positive terms that vary widely in magnitude the decreasing ordering may not allow the smaller terms to contribute to the sum (which is why the harmonic sum $\sum_{i=1}^{n} 1/i$ ' "converges" in floating point arithmetic as $n \to \infty$). However, consider the example with $n = 4$ and

$$x = [1, \quad M, \quad 2M, \quad -3M], \tag{4.5}$$

where M is a floating point number so large that $fl(1 + M) = M$ (thus $M > u^{-1}$). The three orderings considered so far produce the following results:

$$\begin{aligned} \text{Increasing:} \quad & \widehat{S}_n = fl(1 + M + 2M - 3M) = 0, \\ \text{Psum:} \quad & \widehat{S}_n = fl(1 + M - 3M + 2M) = 0, \\ \text{Decreasing:} \quad & \widehat{S}_n = fl(-3M + 2M + M + 1) = 1. \end{aligned}$$

Thus the decreasing ordering sustains no rounding errors and produces the exact answer, while both the increasing and Psum orderings yield computed sums with relative error 1. The reason why the decreasing ordering performs so well in this example is that it adds the "1" after the inevitable heavy cancellation has taken place, rather than before, and so retains the important information in this term. If we evaluate the term $\mu = \sum_{i=2}^{n} |\widehat{S}_i|$ in the error bound (4.3) for example (4.5) we find

$$\text{Increasing: } \mu = 4M, \quad \text{Psum: } \mu = 3M, \quad \text{Decreasing: } \mu = M + 1,$$

so (4.3) "predicts" that the decreasing ordering will produce the most accurate answer, but the bound it provides is extremely pessimistic since there are no rounding errors in this instance.

Extrapolating from this example, we conclude that the decreasing ordering is likely to yield greater accuracy than the increasing or Psum orderings whenever there is heavy cancellation in the sum, that is, whenever $|\sum_{i=1}^{n} x_i| \ll \sum_{i=1}^{n} |x_i|$.

Turning to the insertion method, a good explanation of the insertion strategy is that it attempts to minimize, one at a time, the terms $|\widehat{T}_1|, \dots, |\widehat{T}_{n-1}|$ in the error bound (4.3). Indeed, if the x_i are all nonnegative the insertion method minimizes this bound over all instances of Algorithm 4.1.

Finally, we note that a stronger form of the bound (4.4) holds for pairwise summation. It can be deduced from (4.3) or derived directly, as follows. Assume for simplicity that $n = 2^r$. Unlike in recursive summation each addend takes part in the same number of additions, $\log_2 n$. Therefore we have a relation of the form

$$\widehat{S}_n = \sum_{i=1}^{n} x_i \prod_{k=1}^{\log_2 n} (1 + \delta_k^{(i)}), \qquad |\delta_k^{(i)})| \le u,$$

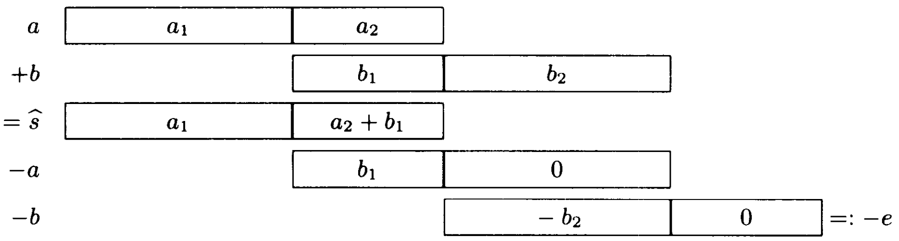


Figure 4.1. *Recovering the rounding error.*

which leads to the bound

$$|E_n| \le \gamma_{\log_2 n} \sum_{i=1}^{n} |x_i|. \tag{4.6}$$

Since it is proportional to $\log_2 n$ rather than n, this is a smaller bound than (4.4), which is the best bound of this form that holds in general for Algorithm 4.1.

4.3. Compensated Summation

We have left to last the *compensated summation* method, which is recursive summation with a correction term cleverly designed to diminish the rounding errors. Compensated summation is worth considering whenever an accurate sum is required and computations are already taking place at the highest precision supported by the hardware or the programming language in use.

In 1951 Gill [449, 1951] noticed that the rounding error in the sum of two numbers could be estimated by subtracting one of the numbers from the sum, and he made use of this estimate in a Runge-Kutta code in a program library for the EDSAC computer. Gill's estimate is valid for fixed point arithmetic only. Kahan [625, 1965] and Møller [777, 1965] both extended the idea to floating point arithmetic. Møller shows how to estimate $a + b - fl(a + b)$ in chopped arithmetic, while Kahan uses a slightly simpler estimate to derive the compensated summation method for computing $\sum_{i=1}^{n} x_i$.

The estimate used by Kahan is perhaps best explained with the aid of a diagram. Let a and b be floating point numbers with $|a| \ge |b|$, let $\widehat{s} = fl(a+b)$, and consider Figure 4.1, which uses boxes to represent the mantissas of a and b. The figure suggests that if we evaluate

$$e = -\big[\big((a+b)-a\big)-b\big] = (a-\widehat{s})+b$$

in floating point arithmetic, in the order indicated by the parentheses, then the computed ê will be a good estimate of the error $(a + b) - \widehat{s}$. In fact, for rounded floating point arithmetic in base 2, we have

$$a+b = \widehat{s} + \hat{e}, \tag{4.7}$$

that is, the computed $\hat{e}$ represents the error exactly. This result (which does not hold for all bases) is proved by Dekker [275, 1971, Thm. 4.7], Knuth [668, 1981, Thm. C, p. 221], and Linnainmaa [703, 1974, Thm. 3]. Note that there is no point in computing $fl(\widehat{s} + \hat{e})$, since $\widehat{s}$ is already the best floating point representation of $a + b$!

Kahan's compensated summation method employs the correction e on every step of a recursive summation. After each partial sum is formed, the correction is computed and immediately added to the next term x_i before that term is added to the partial sum. Thus the idea is to capture the rounding errors and feed them back into the summation. The method may be written as follows.

Algorithm 4.2 (compensated summation). Given floating point numbers $x_1, \ldots, x_n$ this algorithm forms the sum $= \sum_{i=1}^{n} x_i$ by compensated summation.

```
$s = 0$; $e = 0$
for $i = 1:n$
    $temp = s$
    $y = x_i + e$
    $s = temp + y$
    $e = (temp - s) + y$  % Evaluate in the order shown.
end
```

The compensated summation method has two weaknesses: $\hat{e}$ is not necessarily the exact correction, since (4.7) is based on the assumption that $|a| \geq |b|$, and the addition $y = x_i + e$ is not performed exactly. Nevertheless, the use of the corrections brings a benefit in the form of an improved error bound. Knuth [668, 1981, Ex. 19, pp. 229, 572-573] shows that the computed sum $\widehat{S}_n$ satisfies

$$\widehat{S}_n = \sum_{i=1}^{n}(1+\mu_i)x_i, \qquad |\mu_i| \leq 2u + O(nu^2), \tag{4.8}$$

which is an almost ideal backward error result (a more detailed version of Knuth's proof is given by Goldberg [457, 1991]).

In [627, 1972] and [628, 1973] Kahan describes a variation of compensated summation in which the final sum is also corrected (thus "$s = s + e$" is appended to the algorithm above). Kahan states in [627, 1972] and proves in

[628, 1973] that (4.8) holds with the stronger bound $|\mu_i| \leq 2u+O((n-i+1)u^2)$ The proofs of (4.8) given by Knuth and Kahan are similar; they use the model (2.4) with a subtle induction and some intricate algebraic manipulation.

The forward error bound corresponding to (4.8) is

$$|E_n| \leq \bigl(2u + O(nu^2)\bigr) \sum_{i=1}^{n} |x_i|. \tag{4.9}$$

As long as $nu \leq 1$, the constant in this bound is independent of n, and so the bound is a significant improvement over the bounds (4.4) for recursive summation and (4.6) for pairwise summation. Note, however, that if $\sum_{i=1}^n |x_i| \gg |\sum_{i=1}^n x_i|$, compensated summation is not guaranteed to yield a small relative error.

Another version of compensated summation has been investigated by several authors: Jankowski, Smoktunowicz, and Wozniakowski [609, 1983], Jankowski and Wozniakowski [611, 1985], Kielbasinski [655, 1973], Neumaier [788, 1974], and Nickel [797, 1970]. Here, instead of immediately feeding each correction back into the summation, the corrections are accumulated separately by recursive summation and then the global correction is added to the computed sum. For this version of compensated summation Kielbasinski [655, 1973] and Neumaier [788, 1974] show that

$$\widehat{S}_n = \sum (1 + \mu_i) x_i, \qquad |\mu_i| \leq 2u + n^2u^2, \tag{4.10}$$

provided $nu \leq 0.1$; this is weaker than (4.8) in that the second-order term has an extra factor n. If $n^2u \leq 0.1$ then in (4.10), $|\mu_i| \leq 2.1u$. Jankowski, Smoktunowicz, and Wozniakowski [609, 1983] show that, by using a divide and conquer implementation of compensated summation, the range of n for which $|\mu_i| \leq cu$ holds in (4.10) can be extended, at the cost of a slight increase in the size of the constant c.

Neither the correction formula (4.7) nor the result (4.8) for compensated summation holds under the no-guard-digit model of floating point arithmetic. Indeed, Kahan [634, 1990] constructs an example where compensated summation fails to achieve (4.9) on certain Cray machines, but he states that such failure is extremely rare. In [627, 1972] and [628, 1973] Kahan gives a modification of the compensated summation algorithm in which the assignment "$e = (temp - s) + y$" is replaced by

```
f = 0
if sign(temp) = sign(y), f = (0.46 * s - s) + s, end
e = ((temp - f) - (s - f)) + y
```

Kahan shows in [628, 1973] that the modified algorithm achieves (4.8) “on all North American machines with floating hardware” and explains that “The mysterious constant 0.46, which could perhaps be any number between 0.25 and 0.50, and the fact that the proof requires a consideration of known machines designs, indicate that this algorithm is not an advance in computer science.”

Viten’ko [1056, 1968] shows that under the no-guard-digit model (2.6) the summation method with the optimal error bound (in a certain sense defined in [1056, 1968]) is pairwise summation. This does not contradict Kahan’s result because Kahan uses properties of the floating point arithmetic beyond those in the no-guard-digit model.

A good illustration of the benefits of compensated summation is provided by Euler’s method for the ordinary differential equation initial value problem $y' = f(x,y)$, $y(a)$ given, which generates an approximate solution according to $y_{k+1} = y_k + hf_k$, $y_0 = y(a)$. We solved the equation $y' = -y$ with $y(0) = 1$ over $[0,1]$ using n steps of Euler’s method ($nh = 1$), with n ranging from 10 to 10^8. With compensated summation we replace the statements $x = x + h$, $y = y + h * f(x,y)$ by (with the initialization $cx = 0$, $cy = 0$)

```
dx = h + cx
new-x = x + dx
cx = (x - new-x) + dx
x = new_x

dy = h * f(x,y) + cy
new-y = y + dy
cy = (y - new-y) + dy
y = new-y
```

Figure 4.2 shows the errors $e_n = |y(1) - \widehat{y}_n|$ where $\widehat{y}_n$ is the computed approximation to $y(1)$. The computations were done in Fortran 90 in single precision arithmetic on a Sun SPARCstation ($u \approx 6 \times 10^{-8}$). Since Euler’s method has global error of order h, the error curve on the plot should be approximately a straight line. For the standard. implementation of Euler’s method the errors e_n start to increase steadily beyond $n = 20{,}000$ because of the influence of rounding errors. With compensated summation the errors e_n are much less affected by rounding errors and do not grow in the range of n shown (for $n = 10^8$, e_n is about 10 times larger than it would be in exact arithmetic). Plots of U-shaped curves showing total error against stepsize are common in numerical analysis textbooks (see, e.g., Forsythe, Malcolm, and Moler [395, 1977, p. 119] and Shampine [910, 1994, p. 259]), but the textbooks rarely point out that the “U” can be flattened out by compensated summation.

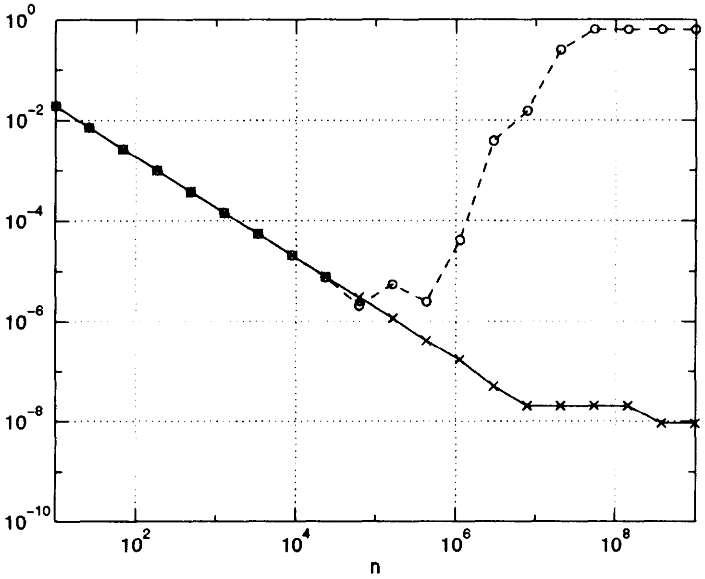


Figure *4.2. Errors* $|y(1) - \widehat{y}_n|$ *for Euler's method with* ("×") *and without* ("o") *compensated summation.*

The cost of applying compensated summation in an ordinary differential equation solver is almost negligible if the function f is at all expensive to evaluate. But, of course, the benefits it brings are noticeable only when a vast number of integration steps are taken. Very long-term integrations are undertaken in celestial mechanics, where roundoff can affect the ability to track planetary orbits. Researchers in astronomy use compensated summation, and other techniques, to combat roundoff. An example application is a 3 million year integration of the planets in the solar system by Quinn, Tremaine, and Duncan [854, 1991]; it used a linear multistep method of order 13 with a constant stepsize of 0.75 days and took 65 days of machine time on a Silicon Graphics 4D-25 workstation. See also Quinn and Tremaine [853, 1991] and Quinlan [851, 1994].

Finally, we describe an even more ingenious algorithm called *doubly compensated summation,* derived by Priest [844, 1992] from a related algorithm of Kahan. It is compensated summation with 2 extra applications of the correction process[11] and it requires 10 instead of 4 additions per step. The algorithm is tantamount to simulating double precision arithmetic with single precision arithmetic; it requires that the summands first be sorted into decreasing order, which removes the need for certain logical tests that would otherwise be necessary.

---

[11] The algorithm should perhaps be called triply compensated summation, but we adopt Priest's terminology.

Algorithm 4.3 (doubly compensated summation). Given floating point numbers $x_1, . . . , x_n$, this algorithm forms the sum $s_n = \sum_{i=1}^{n} x_i$ by doubly compensated summation. All expressions should be evaluated in the order specified by the parentheses.

```
Sort the $x_i$ so that $|x_i| \ge |x_2| \ge . . . \ge |x_n|$.
$s_1 = x_1$; $c_1 = 0$
for $i = 2:n$
    $y_k = c_{k-1} + x_k$
    $u_k = x_k - (y_k - c_{k-1})$
    $t_k = y_k + s_{k-1}$
    $v_k = y_k - (t_k - s_{k-1})$
    $z_k = u_k + v_k$
    $s_k = t_k + z_k$
    $c_k = z_k - (s_k - t_k)$
end
```

Priest [844, 1992, §4.1] analyses this algorithm for t-digit base b arithmetic that satisfies certain reasonable assumptions-ones which are all satisfied by IEEE arithmetic. He shows that if $n \le b^{t-3}$ then the computed sum $\widehat{s}_n$ satisfies

$$|s_n - \widehat{s}_n| \le 2u|s_n|,$$

that is, the computed sum is accurate virtually to full precision.

4.4. Other Summation Methods

We mention briefly two further classes of summation algorithms. The first builds the sum in a series of accumulators, which are themselves added to give the sum. As originally described by Wolfe [1107, 1964] each accumulator holds a partial sum lying in a different interval. Each term x_i is added to the lowest-level accumulator; if that accumulator overflows it is added to the next-highest one and then reset to zero, and this cascade continues until no overflow occurs. Modifications of Wolfe's algorithm are presented by Malcolm [723, 1971] and Ross [880, 1965]. Malcolm [723, 1971] gives a detailed error analysis to show that his method achieves a relative error of order u. A drawback of the algorithm is that it is strongly machine dependent. An interesting and crucial feature of Malcolm's algorithm is that on the final step the accumulators are summed by recursive summation in order of *decreasing* absolute value, which in this particular situation precludes severe loss of significant digits and guarantees a small relative error.

Another class of algorithms, referred to as “distillation algorithms” by Kahan [633, 1987], work as follows: given $x_i = fl(x_i)$, $i = 1{:}n$, they iteratively construct floating point numbers $x_1^{(k)}, \dots, x_n^{(k)}$ such that $\sum_{i=1}^n x_i^{(k)} = \sum_{i=1}^n x_i$, terminating when $x_n^{(k)}$approximates $\sum_{i=1}^n x_i$ with relative error at most u. Kahan states that these algorithms appear to have average run times of order at least n log n. See Bohlender [130, 1977], Kahan [633, 1987], Leuprecht and Oberaigner [700, 1982], Pichat [830, 1972], and Priest [844, 1992, pp. 66-69) for further details and references.

4.5. Statistical Estimates of Accuracy

The rounding error bounds presented above can be very pessimistic, because they account for the worst-case propagation of errors. An alternative way to compare summation methods is through statistical estimates of the error, which may be more representative of the average case. A statistical analysis of three summation methods has been given by Robertazzi and Schwartz [874, 1988] for the case of nonnegative x_i. They assume that the relative errors in floating point addition are statistically independent and have zero mean and finite variance σ^2. Two distributions of nonnegative x_i are considered: the uniform distribution on $[0,2\mu]$. and the exponential distribution with mean μ. Making various simplifying assumptions Robertazzi and Schwartz estimate the mean square error (that is, the variance of the absolute error) of the computed sums from recursive summation with random, increasing and decreasing orderings, and from insertion summation and pair-wise summation (with the increasing ordering). Their results for the summation of n numbers are given in Table -1.1.

The results show that for recursive summation the ordering affects only the constant in the mean square error, with the increasing ordering having the smallest constant and the decreasing ordering the largest; since the x_i are nonnegative, this is precisely the ranking given by the rounding error bound (4.3). The insertion and pairwise summation methods have mean square errors proportional to n^2 rather than n^3 for recursive summation, and the insertion method has a smaller constant than pairwise summation. This is also consistent with the rounding error analysis, in which for nonnegative x_i the insertion method satisfies an error bound no larger than pairwise summation and the latter method has an error bound with a smaller constant than for recursive summation ($\log_2 n$ versus n).

4.6. Choice of Method

There is a wide variety of summation methods to choose from. For each method the error can vary greatly with the data, within the freedom afforded

Table 4.1. *Mean square errors for nonnegative x_i.*

Distrib.	Increasing	Random	Decreasing	Insertion	Pairwise
Unif(0,2μ)	$0.20\mu^2 n^3\sigma^2$	$0.33\mu^2 n^3\sigma^2$	$0.53\mu^2 n^3\sigma^2$	$2.6\mu^2 n^2\sigma^2$	$2.7\mu^2 n^2\sigma^2$
EXP(μ)	$0.13\mu^2 n^3\sigma^2$	$0.33\mu^2 n^3\sigma^2$	$0.63\mu^2 n^3\sigma^2$	$2.6\mu^2 n^2\sigma^2$	$4.0\mu^2 n^2\sigma^2$

by the error bounds; numerical experiments show that, given any two of the methods, data can be found for which either method is more accurate than the other [553, 1993]. However, some specific advice on the choice of method can be given.

1. If high accuracy is important, consider implementing recursive summation in higher precision; if feasible this may be less expensive (and more accurate) than using one of the alternative methods at the working precision. What can be said about the accuracy of the sum computed at higher precision? If $S_n = \sum_{i=1}^{n} x_i$ is computed by recursive summation at double precision (unit roundoff u^2) and then rounded to single precision, an error bound of the form $|S_n - \widehat{S}_n| \le u|\widehat{S}_n| + nu^2 \sum_{i=1}^{n} |x_i|$ holds. Hence a relative error of order u is guaranteed if $nu \sum_{i=1}^{n} |x_i| \le |S_n|$. Priest [844, 1992, pp. 62-63] shows that if the x_i are sorted in decreasing order of magnitude before being summed in double precision, then $|S_n - \widehat{S}_n| \le 2u|S_n|$ holds provided only that $n \le b^{t-3}$ for t-digit base b arithmetic satisfying certain reasonable assumptions. Therefore the decreasing ordering may be worth arranging if there is a lot of cancellation in the sum. An alternative to extra precision computation is doubly compensated summation, which is the only other method described here that guarantees a small relative error in the computed sum.

2. For most of the methods the errors are, in the worst case, proportional to n. If n is very large, pairwise summation (error constant $\log_2 n$) and compensated summation (error constant of order 1) are attractive.

3. If the x_i all have the same sign then all the methods yield a relative error of at most nu and compensated summation guarantees perfect relative accuracy (as long as $nu \le 1$). For recursive summation of one-signed data, the increasing ordering has the smallest error bound (4.3) and the insertion method minimizes this error bound over all instances of Algorithm 4.1.

4. For sums with heavy cancellation $(\sum_{i=1}^{n} |x_i| \gg |\sum_{i=1}^{n} x_i|)$, recursive summation with the decreasing ordering is attractive, although it cannot be guaranteed to achieve the best accuracy.

Considerations of computational cost and the way in which the data are generated may rule out some of the methods. Recursive summation in the natural order, pairwise summation, and compensated summation can be implemented in $O(n)$ operations for general x_i, but the other methods are more expensive since they require searching or sorting. Furthermore, in an application such as the numerical solution of ordinary differential equations, where x_k is not known until $\sum_{i=1}^{k-1} x_i$ has been formed. sorting and searching may be impossible.

4.7. Notes and References

This chapter is based on Higham [553, 1993]. Analysis of Algorithm 4.1 and compensated summation can also be found in Espelid [356, 1978].

The earliest error analysis of summation is that of Wilkinson for recursive summation in [1084, 1960], [1088, 1963].

Pairwise summation was first discussed by McCracken and Dorn [743, 1964, pp. 61-63], Babuska [35, 1969], and Linz [707, 1970]. Caprani [185, 1971] shows how to implement the method on a serial machine using temporary storage of size $[\log_2 n] + 1$ (without overwriting the x_i).

The use of compensated summation with a Runge-Kutta formula is described by Vitasek [1055, 1969]. See also Butcher [170, 1987, pp. 118-120] and the experiments of Linnainrnaa [703, 1974]. Davis and Rabinowitz [267, 1984, §4.2.1] discuss pairwise summation and compensated summation in the context of quadrature.

Problems

4.1. Define and evaluate a condition number $C(x)$ for the summation $S_n(x) = \sum_{i=1}^{n} x_i$. When does the condition number take the value 1?

4.2. (Wilkinson [1088, 1963, p. 19]) Show that the bounds (4.3) and (4.4) are nearly attainable for recursive summation. (Hint: assume $u = 2^{-t}$, set $n = 2^r$ ($r \ll t$), and define

$$\begin{aligned} x(1) &= 1, \\ x(2) &= 1 - 2^{-t}, \\ x(3:4) &= 1 - 2^{1-t}, \\ x(5:8) &= 1 - 2^{2-t}, \\ \\ x(2^{r-1} + 1:2^r) &= 1 - 2^{r-1-t}.) \end{aligned}$$

4.3. Let $S_n = \sum_{i=1}^{n} x_i$ be computed by recursive summation in the natural order. Show that

$$\widehat{S}_n = (x_1 + x_2)(1 + \theta_{n-1}) + \sum_{i=3}^{n} x_i(1 + \theta_{n-i+1}), \qquad |\theta_k| \le \gamma_k = \frac{ku}{1-ku},$$

and hence that $E_n = \widehat{S}_n - S_n$ satisfies

$$|E_n| \le (|x_1| + |x_2|)\gamma_{n-1} + \sum_{i=3}^{n} |x_i|\gamma_{n-i+1}.$$

Which ordering of the x_i minimizes this bound?

4.4. Let M be a floating point number so large that $fl(10 + M) = M$. What are the possible values of $fl(\sum_{i=1}^{6} x_i)$, where $\{x_i\}_{i=1}^{6} = \{1, 2, 3, 4, M, -M\}$ and the sum is evaluated by recursive summation?

4.5. The "±" method for computing $S_n = \sum_{i=1}^{n}$ is defined as follows: form the sum of the positive numbers, S_+, and the sum of the nonpositive numbers, S_-, separately, by any method, and then form $S_n = S_- + S_+$. Discuss the pros and cons of this method.

4.6. Let $\{x_i\}$ be a convergent sequence with limit ξ. Aitken's Δ^2-method (Aitken extrapolation) generates a transformed sequence $\{y_i\}$ defined by

$$y_i := x_i - \frac{(x_{i+1} - x_i)^2}{x_{i+2} - 2x_{i+1} + x_i}.$$

Under suitable conditions (typically that $\{x_i\}$ is linearly convergent), the y_i converge to ξ faster than the x_i. Which of the following expressions should be used to evaluate the denominator in the formula for y_i?

(*a*) $(x_i+2 - 2x_{i+1}) + x_i$.

(*b*) $(x_{i+2} - x_{i+1}) - (x_{i+1} - x_i)$.

(*c*) $(x_{i+2} + x_i) - 2x_{i+1}$.

4.7. Analyse the accuracy of the following method for evaluating $S_n = \sum_{i=1}^{n} x_i$:

$$S_n = \log \prod_{i=1}^{n} e^{x_i}.$$

4.8. In numerical methods for quadrature and for solving ordinary differential equation initial value problems it is often necessary to evaluate a function on an equally spaced grid of points on a range $[a,b]$: $x_i := a + ih$, $i = 0{:}n$, where $h = (b-a)/n$. Compare the accuracy of the following ways to form x_i. Assume that a and b, but not necessarily h, are floating point numbers.

(a) $x_i = x_{i-1} + h$ ($x_0 = a$).

(b) $x_i = a + ih$.

(c) $x_i = a(1 - i/n) + (i/n)b$.

Note that (a) is typically used without comment in, for example, a Newton-Cotes quadrature rule or a Runge-Kutta method with fixed step size.

4.9. (RESEARCH PROBLEM) Priest [844, 1992, pp. 61-62] has proved that if $|x_1| \geq |x_2| \geq |x_3|$ then compensated summation computes the sum $x_1 + x_2 + x_3$ with a relative error of order u (under reasonable assumptions on the arithmetic, such as the presence of a guard digit). He also gives the example

$$x_1 = 2^{t+1}, \quad x_2 = 2^{t+1} - 2, \quad x_3 = x_4 = x_5 = x_6 = -(2^t - 1),$$

for which the exact sum is 2 but compensated summation computes 0 in IEEE single precision arithmetic (t=24). What is the smallest n for which compensated summation applied to $x_1, \ldots, x_n$ ordered by decreasing absolute value can produce a computed sum with large relative error?

Chapter 5
Polynomials

The polynomial $(z - 1)(z - 2) \ldots (z - 20)$ *is not a 'difficult' polynomial* per se . . .
The 'difficulty' with the polynomial $P(z - i)$ *is that of evaluating the* explicit *polynomial accurately.*
If one already knows the roots, then the polynomial can be evaluated without any loss of accuracy.

-J. H. WILKINSON, *The Perfidious Polynomial (1984)*

I first used backward error analysis in connection with
simple programs for computing zeros of polynomials
soon after the PILOT ACE came into use.

-J. H. WILKINSON, *The State of the Art in Error Analysis (1985)*

The Fundamental Theorem of Algebra asserts that
every polynomial equation over the complex field has a root.
It is almost beneath the dignity of such a majestic theorem
to mention that in fact it has precisely n roots.

-J. H. WILKINSON, *The Perfidious Polynomial* (1984)

Two common tasks associated with polynomials are evaluation and interpolation: given the polynomial find its values at certain arguments, and given the values at certain arguments find the polynomial. We consider Horner's rule for evaluation and the Newton divided difference polynomial for interpolation. A third task not considered here is finding the zeros of a polynomial. Much research was devoted to polynomial zero finding up until the late 1960s; indeed, Wilkinson devotes a quarter of *Rounding Errors in Algebraic Processes* [1088, 1963] to the topic. Since the development of the QR algorithm for finding matrix eigenvalues there has been less demand for polynomial zero finding, since the problem either arises as, or can be converted to (see §26.6 and [346, 1995], [1007, 1994]), the matrix eigenvalue problem.

5.1. Horner's Method

The standard method for evaluating a polynomial

$$p(x) = a_0 + a_1 x + \dots + a_n x^n \tag{5.1}$$

is Horner's method (also known as Horner's rule and nested multiplication), which consists of the following recurrence:

```
$q_n(x) = a_n$
for $i = n - 1: -1:0$
    $q_i(x) = xq_{i+1}(x) + a_i$
end
$p(x) = q_0(x)$
```

The cost is $2n$ flops, which is n less than the more obvious method of evaluation that explicitly forms powers of x (see Problem 5.2).

To analyse the rounding errors in Horner's method it is convenient to use the relative error counter notation $<k>$ (see (3.9)). We have

$$\begin{aligned}\widehat{q}_{n-1} &= (x\widehat{q}_n<1> + a_{n-1})<1>\\ &= xa_n<2> + a_{n-1}<1>,\\ \widehat{q}_{n-2} &= (x\widehat{q}_{n-1}<1> + a_{n-2})<1>\\ &= x^2a_n<4> + xa_{n-1}<3> + a_{n-2}<1>.\end{aligned}$$

It is easy to either guess or prove by induction that

$$\begin{aligned}\widehat{q}_0 &= a_0<1> + a_1x<3> + \dots + a_{n-1}x^{n-1}<2n-1> + a_nx^n<2n>\\ &= (1+\theta_1)a_0 + (1+\theta_3)a_1x + \dots + (1+\theta_{2n-1})a_{n-1}x^{n-1}\\ &\quad + (1+\theta_{2n})a_nx^n,\end{aligned} \tag{5.2}$$

where we have used Lemma 3.1, and where $|\phi_k| \le ku/(1-ku) =: \gamma_k$. This result shows that Horner's method has a small backward error: the computed $\widehat{q}_0$ is the exact value at x of a polynomial obtained by making relative perturbations of size at most γ_{2n} to the coefficients of $p(x)$.

A forward error bound is easily obtained: from (5.2) we have

$$|p(x) - \widehat{q}_0| \le \gamma_{2n} \sum_{i=0}^{n} |a_i||x|^i = \gamma_{2n}\,\widetilde{p}(|x|), \tag{5.3}$$

where $\widetilde{p}(x) = \sum_{i=0}^{n} |a_i| x^i$. The relative error is bounded according to

$$\frac{|p(x) - \widehat{q}_0|}{|p(x)|} \le \gamma_{2n} \frac{\widetilde{p}(|x|)}{|p(x)|} =: \gamma_{2n}\psi(p,x).$$

Clearly, the factor $\psi(p,x)$ can be arbitrarily large. However, $\psi(p,x) = 1$ if $a_i \ge 0$ for all i and $x \ge 0$, or if $(-1)^i a_i \ge 0$ for all i and $x \le 0$.

In a practical computation we may wish to compute an error bound along with $\widehat{q}_0$. The bound (5.3) is entirely adequate for theoretical purposes and can itself be computed by Horner's method. However, it lacks sharpness for two reasons. First, the bound is the result of replacing each γ_k by γ_{2n}. Second, and more importantly, it is an a priori bound and so takes no account of the actual rounding errors that occur. We can derive a sharper, a posteriori bound by a running error analysis.

For the ith step of Horner's method we can write

$$(1+\epsilon_i)\widehat{q}_i = x\widehat{q}_{i+1}(1+\delta_i) + a_i, \qquad |\delta_i|, |\epsilon_i| \le u, \tag{5.4}$$

where we have used both (2.4) and (2.5). Defining $\widehat{q}_i =: q_i + f_i$, we have

$$q_i + f_i + \epsilon_i \widehat{q}_i = x(q_{i+1} + f_{i+1}) + x\widehat{q}_{i+1}\delta_i + a_i,$$

or

$$f_i = x f_{i+1} + x\widehat{q}_{i+1}\delta_i - \epsilon_i \widehat{q}_i, \quad f_n = 0.$$

Hence

$$|f_i| \le |x||f_{i+1}| + u(|x||\widehat{q}_{i+1}| + |\widehat{q}_i|).$$

Since $f_n = 0$, we have $|f_i| \le u\pi_i$, where

$$\pi_i = |x|\pi_{i+1} + |x||\widehat{q}_{i+1}| + |\widehat{q}_i|, \quad \pi_n = 0.$$

We can slightly reduce the cost of evaluating the majorizing sequence π_i by working with $\mu_i := \frac{1}{2}(\pi_i + |\widehat{q}_i|)$, which satisfies the recurrence

$$\mu_i = |x|\mu_{i+1} + |\widehat{q}_i|, \quad \mu_n = \frac{1}{2}|\widehat{q}_n|.$$

We can now furnish Horner's method with a running error bound.

Algorithm 5.1. This algorithm evaluates $y = fl(p(x))$ by Horner's method, where $p(x) = \sum_{i=0}^{n} a_i x^i$. It also evaluates a quantity μ such that $|y - p(x)| \le \mu$.

```
$y = a_n$
$\mu = |y|/2$
for $i = n-1:-1:0$
    $y = xy + a_i$
    $\mu = |x|\mu + |y|$
end
$\mu = \mu(2\mu - |y|)$
```

Cost: $4n$ flops.

It is worth commenting on the case where one or more of the a_i and x is complex. The analysis leading to Algorithm 5.1 is still valid for complex data, but we need to remember that the error bounds for $fl(x \text{ op } y)$ are not the same as for real arithmetic. In view of Lemma 3.5. it suffices to replace the last line of the algorithm by $\mu = \sqrt{2}\gamma_2(2\mu - |y|)$. An increase in speed of the algorithm, with only a slight worsening of the bound, can be obtained by replacing $|y| = ((\mathrm{Re}\, y)^2 + (\mathrm{Im}\, y)^2)^{1/2}$ by $|\mathrm{Re}\, y| + |\mathrm{Im}\, y|$ (and, of course, $|x|$ should be evaluated once and for all before entering the loop).

One use of Algorithm 5.1 is to provide a stopping criterion for a polynomial zero-finder: if $|fl(p(x))|$ is of the same order as the error bound μ, then further iteration serves no purpose, for as far as we can tell, x could be an exact zero.

As a numerical example, for the expanded form of $p(x) = (x + 1)^{32}$ we found in MATLAB that

$$fl(p(-1)) \equiv 0, \quad \mu = 2.4 \times 10^{-7}, \quad \gamma_{2n}\widetilde{p}(|x|) = 1.5 \times 10^{-5},$$

and for $p(x)$ the Chebyshev polynomial of degree 32,

$$fl(p(0.5)) \approx 0.5000, \quad \mu = 3.3 \times 10^{-10}, \quad \gamma_{2n}\widetilde{p}(|x|) = 1.0 \times 10^{-8}.$$

In these two cases. the running error bound is, respectively. 62 and 31 times smaller than the a priori one.

In another experiment we evaluated the expanded form of $p(x) = (x - 2)^3$ is simulated single precision in MATLAB ($u \approx 6 \times 10^{-8}$) for 200 equally spaced points near $x = 2$. The polynomial values, the error, and the a priori and running error bounds are all plotted in Figure 5.1. The running error bound is about seven times smaller than the a priori one.

5.2. Evaluating Derivatives

Suppose now that we wish to evaluate derivatives of p. We could simply differentiate (5.1) as many times as necessary and apply Horner's method to

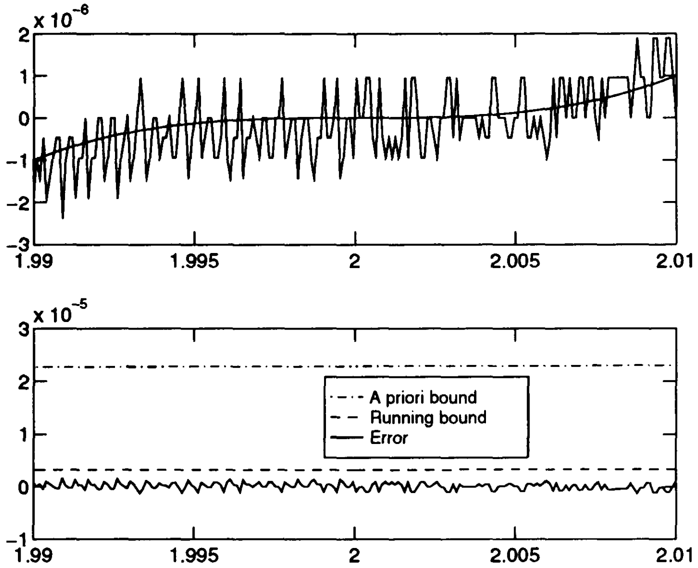


Figure 5.1. *Computed polynomial values (top) and running and a priori bounds (bottom) for Horner's method.*

each expression, but there is a more efficient way. Observe that if we define

$$q(x) = q_1 + q_2 x + \dots + q_n x^{n-1} \quad r = q_0,$$

where the $q_i = q_i(a)$ are generated by Horner's method for $p(a)$, then

$$p(x) = (x - a)q(x) + r.$$

In other words, Horner's method carries out the process of synthetic division. Clearly, $p'(a) = q(a)$. If we repeat synthetic division recursively on $q(x)$, we will be evaluating the coefficients in the Taylor expansion

$$p(x) = p(\alpha) + (x-\alpha)p'(\alpha) + \frac{(x-\alpha)^2}{2!}p''(\alpha) + \cdots + \frac{(x-\alpha)^n}{n!}p^{(n)}(\alpha),$$

and after a final scaling by factorials, we will obtain the derivatives of p at a. The resulting algorithm is quite short.

Algorithm 5.2. This algorithm evaluates the polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ and its first k derivatives at a, returning $y_i = p^{(i)}(a)$, $i = 0{:}k$.

```
$y_0 = a_n$
$y(1:k) = 0$
```

```
for $j = n-1: -1: 0$
    for $i = \min(k, n-j): -1:1$
        $y_i = \alpha y_i + y_{i-1}$
    end
    $y_0 = \alpha y_0 + a_j$
end
$m = 1$
for $j = 2:k$
    $m = m*j$
    $y_j = m*y_j$
end
```

cost: $nk + 2(k + n) - k^2/2$ flops.

How is the error bounded for the derivatives in Algorithm 5.2? To answer this quest ion with the minimum of algebra, we express the algorithm in matrix notation. Horner's method for evaluating $p(\alpha)$ is equivalent to solution of the bidiagonal system

$$U_{n+1}q := \begin{bmatrix} 1 & -\alpha & & & \\ & 1 & -\alpha & & \\ & & 1 & \ddots & \\ & & & \ddots & -\alpha \\ & & & & 1 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ \vdots \\ q_n \end{bmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ \vdots \\ a_n \end{bmatrix} =: a.$$

By considering (5.4). we see that

$$(U_{n+1} + \Delta_1)\widehat{q} = a, \qquad |\Delta_1| \le u|U_{n+1}|.$$

Hence

$$|q - \widehat{q}| \le u|U_{n+1}^{-1}||U_{n+1}||q| + O(u^2). \tag{5.5}$$

The recurrence for $r_0 = p'(\alpha)$ can be expressed as $U_n r = q(1:n)$, where $r = r(0:n-1)$, so

$$(U_n + \Delta_2)\widehat{r} = \widehat{q}(1:n), \qquad |\Delta_2| \le u|U_n|.$$

Hence

$$\begin{aligned}\widehat{r} &= (U_n^{-1} - U_n^{-1}\Delta_2 U_n^{-1})\big(q(1:n) + (\widehat{q}(1:n) - q(1:n))\big) + O(u^2) \\ &= r - U_n^{-1}\Delta_2 r + U_n^{-1}(\widehat{q}(1:n) - q(1:n)) + O(u^2).\end{aligned}$$

This gives, using (5.5),

$$|r - \widehat{r}| \le u|U_n^{-1}||U_n||r| + u|U_n^{-1}||U_n^{-1}||U_n||q(1:n)| + O(u^2). \tag{5.6}$$

Now

$$|U_n^{-1}| = \begin{bmatrix} 1 & |\alpha| & |\alpha|^2 & \dots & |\alpha|^{n-1} \\ & 1 & |\alpha| & & \vdots \\ & & 1 & \ddots & \vdots \\ & & & \ddots & |\alpha| \\ & & & & 1 \end{bmatrix},$$

$$|U_n^{-1}||U_n| = \begin{bmatrix} 1 & 2|\alpha| & 2|\alpha|^2 & \dots & 2|\alpha|^{n-1} \\ & 1 & 2|\alpha| & & \vdots \\ & & 1 & \ddots & \vdots \\ & & & \ddots & 2|\alpha| \\ & & & & 1 \end{bmatrix},$$

$$|U_n^{-1}||U_n^{-1}||U_n| = \begin{bmatrix} 1 & 3|\alpha| & 5|\alpha|^2 & \dots & (2n-1)|\alpha|^{n-1} \\ & 1 & 3|\alpha| & & \vdots \\ & & 1 & \ddots & \vdots \\ & & & \ddots & 3|\alpha| \\ & & & & 1 \end{bmatrix}.$$

By looking at the form of r and q, we find from (5.6) that

$$\begin{aligned} |p'(\alpha) - \widehat{r}_0| &\le 2u \sum_{k=1}^{n} k^2 |a_k||\alpha|^{k-1} + O(u^2) \\ &\le 2nu \sum_{k=1}^{n} k|a_k||\alpha|^{k-1} + O(u^2) \\ &=: 2nu\,\widetilde{p}'(\alpha) + O(u^2). \end{aligned} \tag{5.7}$$

This is essentially the same form of bound as for $p(a)$ in (5.3). Analogous bounds hold for all derivatives.

5.3. The Newton Form and Polynomial Interpolation

An alternative to the monomial representation of a polynomial is the Newton form

$$p(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (x - \alpha_j), \tag{5.8}$$

which is commonly used for polynomial interpolation. The interpolation problem is to choose p so that $p(a_i) = f_i$, $i = 0{:}n$, and the numbers c_i are known as divided differences. Assuming that the points a_j are distinct, the divided differences may be computed from a standard recurrence:

```
$c^{(0)}(0:n) = f(0:n)$
for $k = 0:n-1$
    for $j = n:-1:k+1$
        $c_j^{(k+1)} = (c_j^{(k)} - c_{j-1}^{(k)})/(\alpha_j - \alpha_{j-k-1})$
    end
end
$c = c^{(n)}$
```

Cost: $3n^2/2$ flops.

Two questions are of interest: how accurate are the computed $\widehat{c}_j$, and what is the effect of rounding errors on the polynomial values obtained by evaluating the Newton form? To answer the first question we express the recurrence in matrix-vector form:

$$c^{(0)} = f, \qquad c^{(k+1)} = L_k c^{(k)}, \quad k = 0:n-1,$$

where $L_k = D_k^{-1} M_k$ is lower bidiagonal, with

$$D_k = \mathrm{diag}(\mathrm{ones}(1:k+1),\ a_{k+1} - a_0, a_{k+2} - a_1, \ldots, a_n - a_{n-k-1}),$$

$$M_k = \begin{bmatrix} I_k & & & & & \\ & 1 & & & & \\ & -1 & 1 & & & \\ & & -1 & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & -1 & 1 \end{bmatrix}.$$

The analysis that follows is based on the model (2.4), and so is valid only for machines with a guard digit. With the no-guard-digit model (2.6) the bounds become weaker and more complicated, because of the importance of terms $fl(a_j - a_{j-k-1})$ in the analysis.

It is straightforward to show that

$$\widehat{c}^{(k+1)} = G_k L_k \widehat{c}^{(k)}, \tag{5.9}$$

where $G_k = \mathrm{diag}(\mathrm{ones}(1:k+1), h_{k,k+2}, \ldots, h_{k,n+1})$, where each h_{ij} is of the form $h_{ij} = (1+\delta_1)(1+\delta_2)(1+\delta_3)$, $|\delta_i| \le u$. Hence

$$\widehat{c} = (L_{n-1} + \Delta L_{n-1}) \ldots (L_0 + \Delta L_0) f, \qquad |\Delta L_i| \le \gamma_3 |L_i|. \tag{5.10}$$

From Lemma 3.7.

$$\begin{aligned} |c - \widehat{c}| &= \big|\big(L_{n-1} \ldots L_0 - (L_{n-1} + \Delta L_{n-1}) \ldots (L_0 + \Delta L_0)\big) f\big| \\ &\le \big((1+\gamma_3)^n - 1\big)|L_{n-1}| \ldots |L_0||f| \\ &= \big((1-3u)^{-n} - 1\big)|L_{n-1}| \ldots |L_0||f|. \end{aligned} \tag{5.11}$$

To interpret the bound, note first that merely rounding the data ($f_i \rightarrow f_i(1+\delta_i)$, $|\delta_i| \le u$) can cause an error Δc as large as $e_{\text{round}} = u|L||f|$, where $L = L_{n-1} \dots L_0$, so errors of at least this size are inevitable. Since $|L_{n-1}| \dots |L_0| \ge |L_{n-1} \dots L_0| = |L|$, the error in the computed divided differences can be larger than e_{round} only if there is much subtractive cancellation in the product $L = L_{n-1} \dots L_0$. If $a_0 < a_1 < \dots < a_n$, then each L_i is positive on the diagonal and nonpositive on the first subdiagonal; therefore $|L_{n-1}| \dots |L_0| = |L_{n-1} \dots L_0| = |L|$, and we have the very satisfactory bound $|c - \widehat{c}| \le ((1-3u)^{-n} - 1)|L||f|$. This same bound holds if the a_i are arranged in decreasing order.

To examine how well the computed Newton form reproduces the f_i we "unwind" the analysis above. From (5.9) we have

$$\widehat{c}^{(k)} = L_k^{-1} G_k^{-1} \widehat{c}^{(k+1)} = (L_k^{-1} + \widetilde{\Delta L_k}) \widehat{c}^{(k+1)}, \qquad |\widetilde{\Delta L_k}| \le \gamma_3 |L_k^{-1}|.$$

By invoking Lemma 3.7 again, we obtain

$$|f - L^{-1}\widehat{c}| \le \big((1-3u)^{-n} - 1\big) |L_0^{-1}| \dots |L_{n-1}^{-1}| |\widehat{c}|. \tag{5.12}$$

If $a_0 < a_1 < \dots < a_n$ then $L_i^{-1} \ge 0$ for all i, and we obtain the very satisfactory bound $|f - L^{-1}\widehat{c}| \le ((1-3u)^{-n} - 1)|L^{-1}||\widehat{c}|$. Again, the same bound holds for points arranged in decreasing order.

In practice it is found that even when the computed divided differences are very inaccurate, the computed interpolating polynomial may still reproduce the original data well. The bounds (5.11) and (5.12) provide insight into this observed behaviour by showing that $c - \widehat{c}$ and $f - L^{-1}\widehat{c}$ can be large only when there is much cancellation in the products $L_{n-1} \dots L_0 f$ and $L_0^{-1} \dots L_{n-1}^{-1} c$, respectively.

The analysis has shown that the ordering $a_0 < a_1 < \dots < a_n$ yields "optimal" error bounds for the divided differences and the residual, and so may be a good choice of ordering of interpolation points. However, if the aim is to minimize $|p(x) - fl(p(x))|$ for a given $x \ne a_j$, then other orderings need to be considered. An ordering with some theoretical support is the Leja ordering, which is defined by the equations [863, 1990]

$$\alpha_0 = \max_i |\alpha_i|, \tag{5.13a}$$

$$\prod_{k=0}^{j-1} |\alpha_j - \alpha_k| = \max_{i \ge j} \prod_{k=0}^{j-1} |\alpha_i - \alpha_k|, \quad j = 1{:}\,n-1. \tag{5.13b}$$

For a given set of $n+1$ points a_i, the Leja ordering can be computed in n^2 flops (see Problem 5.4).

We give a numerical example to illustrate the analysis. Let $n = 16$ and let $a_0 < \dots < a_n$ be equally spaced points on [-1,1] . Working in simulated single precision with $u = 2^{-24} \approx 6 \times 10^{-8}$, we computed divided differences for two different vectors f. Error statistics were computed by regarding the solutions computed in double precision as exact. We define the ratios

$$\rho_1 = \max_i \frac{(|L_{n-1}| \dots |L_0||f|)_i}{|c_i|}, \quad \rho_2 = \max_i \frac{(|L_0^{-1}| \dots |L_{n-1}^{-1}||\widehat{c}|)_i}{|f_i|}.$$

(1) For f_i from the normal $N(0,1)$ distribution the divided differences range in magnitude from 1 to 10^5. and their relative errors range from 0 (the first divided difference, f_0, is always exact) to 3×10^{-7}. The ratio $\rho_1 = 16.3$, so (5.11) provides a reasonably sharp bound for the error in $\widehat{c}$.. The relative errors when f is reconstructed from the computed divided differences range between 0 and 3×10^{-1} (it makes little difference whether the reconstruction is done in single or double precision). Again, this is predicted by the analysis, in this case by (5.12), because $\rho_2 = 2 \times 10^7$. For the Leja ordering, the divided differences are computed with about the same accuracy, but f is reconstructed much more accurately, with maximum relative error 7×10^{-6} ($\rho_1 = 1 \times 10^3$, $\rho_2 = 8 \times 10^4$).

(2) For $f_i = \exp(a_i)$, the situation is reversed: we obtain inaccurate divided differences but an accurate reconstruction of f. The divided differences range in magnitude from 10^{-4} to 10^{-1}, and their relative errors are as large as 1, but the relative errors in the reconstructed f are all less than 10^{-7}. Again, the error bounds predict this behaviour: $\rho_1 = 6 \times 10^8$, $\rho_2 = 1.02$. The Leja ordering performs similarly.

The natural way to evaluate the polynomial (5.8) for a given x is by a generalization of Horner's method:

```
$q_n(x) = c_n$
for $i = n - 1:-1:0$
    $q_i(x) = (x-a_i)q_i+1(x) + c_i$
end
$p(x) = q_0(x)$
```

A straightforward analysis shows that (cf. (5.2))

$$\begin{aligned}\widehat{q}_0 = c_0\langle 1\rangle &+ (x-\alpha_0)c_1\langle 4\rangle + (x-\alpha_0)(x-\alpha_1)c_2\langle 7\rangle + \cdots \\ &+ (x-\alpha_0)\dots(x-\alpha_{n-2})c_{n-1}\langle 3n-2\rangle \\ &+ (x-\alpha_0)\dots(x-\alpha_{n-1})c_n\langle 3n\rangle.\end{aligned}$$

Hence the computed $\widehat{q}_0$ is the exact value corresponding to a polynomial with slightly perturbed divided differences. The corresponding forward error bound is

$$|p(x) - \widehat{q}_0| \le \gamma_{3n} \sum_{i=0}^{n} |c_i| \prod_{i=0}^{i-1} |x - a_j|.$$

5.4. Notes and References

Backward and forward error analysis for Horner's rule was given by Wilkinson [1088, 1963, pp. 36-37, 49-50]; our results are simply Wilkinson's presented in a different notation. The analysis has been redone by many other authors, sometimes without reference to Wilkinson's results. Another early reference, which gives a forward error bound only, is McCracken and Dorn [743, 1964, §3.5],

For more on running error analysis see §3.3.

Müller [782, 1983] gives a first-order error analysis for the evaluation of the divided difference form of a polynomial. Olver [809, 1986] derives a posteriori error bounds for the Horner scheme with derivatives (Algorithm 5.2), phrasing them in terms of his relative precision notation. Stewart [939, 1971] analyses synthetic division, using a matrix-oriented approach similar to that in §5.2.

The relative merits of the monomial and Chebyshev representations of a polynomial are investigated, with respect to accuracy of evaluation, by Newbery [793, 1974] and Schonfelder and Razaz [901, 1980]. Clenshaw [212, 1955] showed how Horner's method could be extended to evaluate a polynomial expressed in the Chebyshev form $p(x) = \sum_{i=0}^{n} a_i T_i(x)$, where T_i is the Chebyshev polynomial of degree i. Error analysis of Clenshaw's method, and variations of it, are given by Gentleman [433, 1969], Newbery [792, 1973], and Oliver [805, 1977], [806, 1979]. Clenshaw's scheme can be generalized to expansions in terms of arbitrary orthogonal polynomials; see Smith [927, 1965] and Algorithm 21.8.

Running error bounds for Horner's method were included in algorithms of Kahan and Farkas [637, 1963], [638, 1963] without explanation. Adams [5, 1967] derives the bounds and extends them to evaluation of a real polynomial at a complex argument. Algorithm 5.1 is given in [5, 1967], and also in the classic paper by Peters and Wilkinson [827, 1971], which describes many aspects of the solution of polynomial equations. Wilkinson's paper "The Perfidious Polynomial" [1103, 1984] (for which he was awarded the Chauvenet Prize) is highly recommended as a beautifully written introduction to backward error analysis in general and error analysis for polynomials in particular.

There seems to be little work on choosing the ordering of interpolation points to minimize the effect of rounding errors on the construction or evaluation of the interpolating polynomial. Werner [1075, 1984] examines experimentally the effect of different orderings on the computed value of an interpolating polynomial at a single point, for several forms of interpolating polynomial.

The Leja ordering, which was proposed by Leja in a 1957 paper, is analysed in detail by Reichel [863, 1990]. He shows that an appropriately defined condition number for the Newton form of interpolating polynomial grows at a slower than exponential rate in the degree n for Leja points, which are points taken from a given compact set that satisfy the condition (5.13). For more on the numerical benefits of the Leja ordering see §21.3.3.

If a polynomial is to be evaluated many times at different, arguments it may be worthwhile to expend some effort transforming it to a form that can be evaluated more cheaply than by a straightforward application of Horner's rule. For example, the quartic

$$p(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0, \qquad a_4 \neq 0,$$

can be rewritten as [668, 1981, p. 471]

$$p(x) = ((y + x + a_2)y + a_3)a_4, \qquad y = (x + a_0)x + a_1,$$

where the coefficients a_i are given by.

$$\alpha_0 = \tfrac{1}{2}(a_3/a_4 - 1), \quad \beta = a_2/a_4 - \alpha_0(\alpha_0 + 1), \quad \alpha_1 = a_1/a_4 - \alpha_0\beta,$$
$$\alpha_2 = \beta - 2\alpha_1, \quad \alpha_3 = a_0/a_4 - \alpha_1(\alpha_1 + \alpha_2), \quad \alpha_4 = a_4.$$

Once the a_i have been computed, $p(x)$ can he evaluated in three multiplications and five additions, as compared with the four multiplications and four additions required by Horner's rule. If a multiplication takes longer than an addition, the transformed polynomial should be cheaper to evaluate. For polynomials of degree $n > 4$ there exist evaluation schemes that require strictly less than the $2n$ total additions and multiplications required by Horner's rule; see Knuth [665, 1962], [668, 1981, pp. 471-475] and Fike [373, 1967] One application in which such schemes have been used is in evaluating polynomial approximations in an elementary function library [412, 1991]. Little seems to be known about the numerical stability of fast polynomial evaluation schemes; see Problem 5.6.

Problems

5.1. Give an alternative derivation of Algorithm 5.2 by differentiating the Horner recurrence and rescaling the iterates.

5.2. Give an error analysis for the following "beginner's" algorithm for evaluating $p(x) = a_0 + a_1x + \dots + a_nx^n$:

```
$q(x) = a_0$; $y = 1$
for $i = 1:n$
    $y = xy$
    $q(x) = q(x) + a_iy$
end
$p(x) = q(x)$
```

5.3. Let $p(x) = a_0 + a_1x + ... + a_nx^n$ and $n = 2m$. Then

$$\begin{aligned} p(x) &= (a_0 + a_2x^2 + ... + a_{2m}x^{2m}) + (a_1x + a_3x^3 + ... + a_{2m-1}x^{2m-1}) \\ &= a_0 + a_2y + ... + a_{2m}y^m + x(a_1 + a_3y + ... + a_{2m-1}y^{m-1}), \end{aligned}$$

where $y = x^2$. Obtain an error bound for $fl(p(x))$ when p is evaluated using this splitting (using Horner's rule on each part).

5.4. Write down an algorithm for computing the Leja ordering (5.13) in n^2 flops.

5.5. If the polynomial $p(x) = \sum_{i=0}^{n} a_i x^i$ has roots $x_1, ..., x_n$, it can be evaluated from the root product form $p(x) = a_n \prod_{i=1}^{n}(x - x_i)$. Give an error analysis for this evaluation.

5.6. (RESEARCH PROBLEM) Investigate the numerical stability of fast polynomial evaluation schemes (see the Notes and References) by both rounding error analysis and numerical experiments. For a brief empirical study see Miller [757, 1975, §10].

Previous Home Next

Chapter 6

Norms

While it is true that all norms are equivalent theoretically,
only a homely *one like the ∞-norm is truly useful numerically.*

—J. H. WILKINSON[12], *Lecture at Stanford University* (1984)

Matrix norms are defined in many different ways in the older literature,
but the favorite was the Euclidean norm of the matrix
considered as a vector in n^2-space.
Wedderburn (1934) *calls this the absolute value of the matrix*
and traces the idea back to Peano in 1887.

—ALSTON S. HOUSEHOLDER,
The Theory of Matrices in Numerical Analysis (1964)

---

[12] Quoted in Fox [403, 1987].

Norms are an indispensable tool in numerical linear algebra. Their ability to compress the mn numbers in an $m \times n$ matrix into a single scalar measure of size enables perturbation results and rounding error analyses to be expressed in a concise and easily interpreted form. In problems that are badly scaled, or contain a structure such as sparsity, it is often better to measure matrices and vectors componentwise. But norms remain a valuable instrument for the error analyst, and in this chapter we describe some of their most useful and interesting properties.

6.1. Vector Norms

A vector norm is a function $\|\cdot\| : \mathbb{C}^n \rightarrow \mathbb{R}$ satisfying the following conditions:

1. $\|x\| \geq 0$ with equality iff $x = 0$.
2. $\|\alpha x\| = |\alpha|\,\|x\|$ for all $\alpha \in \mathbb{C}$, $x \in \mathbb{C}^n$.
3. $\|x+y\| \leq \|x\| + \|y\|$ for all $x, y \in \mathbb{C}^n$ (the triangle inequality).

The three most useful norms in error analysis and in numerical computation are

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \qquad \text{"Manhattan" or "taxi cab" norm,}$$

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2} = (x^*x)^{1/2}, \qquad \text{Euclidean length,}$$

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

These are all special cases of the Holder y-norm:

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}, \qquad p \geq 1.$$

The 2-norm has two properties that make it particularly useful for theoretical purposes. First, it is invariant under unitary transformations, for if $Q^*Q = I$, then $\|Qx\|_2^2 = x^*Q^*Qx = x^*x = \|x\|_2^2$. Second, the 2-norm is differentiable for all x, with gradient vector $\nabla\|x\|_2 = x/\|x\|_2$.

A fundamental inequality for vectors is the Hölder inequality (see, for example, [502, 1967, App. 1])

$$|x^*y| \leq \|x\|_p \|y\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1. \tag{6.1}$$

This is an equality when p, $q > 1$ if the vectors $(|x_i|^p)$ and $(|y_i|^q)$ are linearly dependent and $x_i y_i$ lies on the same ray in the complex plane for all i; equality is also possible when $p = 1$ and $p = \infty$, as is easily verified. The special case with $p = q = 2$ is called the Cauchy-Schwarz inequality:

$$|x^*y| \le \|x\|_2 \|y\|_2.$$

For an arbitrary vector norm $\|\cdot\|$ the *dual* norm is defined by

$$\|x\|_D = \max_{z \ne 0} \frac{|z^*x|}{\|z\|}. \tag{6.2}$$

It follows from the Hölder inequality that the dual of the p-norm is the q-norm, where $p^{-1}+q^{-1} = 1$. The definition of dual norm yields, trivially, the general Hölder inequality $|x^*y| \le \|x\| \, \|y\|_D$. For a proof of the reassuring result that the dual of the dual norm is the original norm (the "duality theorem") see Horn and Johnson [580, 1985, Thm. 5.5.14].

In some analyses we need the vector z *dual* to y, which is defined by the property

$$z^*y = \|z\|_D \|y\| = 1. \tag{6.3}$$

That such a vector z exists is a consequence of the duality theorem (see [580, 1985, Cor. 5.5.15]).

How much two p-norms of a vector can differ is shown by the attainable inequalities [422, 1983, pp. 27-28], [459, 1983, Lem. 1.1]

$$\|x\|_{p_2} \le \|x\|_{p_1} \le n^{\left(\frac{1}{p_1} - \frac{1}{p_2}\right)} \|x\|_{p_2}, \qquad p_1 \le p_2. \tag{6.4}$$

The p-norms have the properties that $\|x\|$ depends only on the absolute value of x, and the norm is an increasing function of the absolute values of the entries of x. These properties are important enough to warrant a definition.

Definition 6.1. A norm on $\mathbb{C}^n$ is

1. *monotone* if $|x| \le |y| \Rightarrow \|x\| \le \|y\|$ for all $x, y \in \mathbb{C}^n$, and
2. *absolute* if $\| \, |x| \, \| = \|x\|$ for all $x \in \mathbb{C}^n$.

The following nonobvious theorem shows that these two properties are equivalent.

Theorem 6.2 (Bauer, Stoer, and Witzgall). *A norm on $\mathbb{C}^n$ is monotone if and only if it is absolute.*

Proof. See Horn and Johnson [580, 1985, Thm. 5.5.10], or Stewart and Sun [954, 1990, Thm. 2.1.3]. □

6.2. Matrix Norms

A matrix norm is a function $\|\cdot\| : \mathbb{C}^{m\times n} \rightarrow \mathbb{R}$ satisfying obvious analogues of the three vector norm properties. The simplest example is the Frobenius norm.

$$\|A\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2\right)^{1/2} = \left(\operatorname{trace}(A^*A)\right)^{1/2}$$

(which is sometimes called the Euclidean norm and denoted $\|\cdot\|_E$).

A very important class of matrix norms are those *subordinate* to vector norms. Given a vector norm on $\mathbb{C}^n$, the corresponding subordinate matrix norm on $\mathbb{C}^{m\times n}$ is defined by

$$\|A\| = \max_{x\neq 0} \frac{\|Ax\|}{\|x\|}, \tag{6.5}$$

or, equivalently,

$$\|A\| = \max_{\|x\|=1} \|Ax\|.$$

(Strictly speaking, this definition uses two different norms: one on $\mathbb{C}^m$ in the numerator of (6.5) and one on $\mathbb{C}^n$ in the denominator. Thus the norm used in the definition is assumed to form a family defined on $\mathbb{C}^s$ for any s.)

For the 1-, 2-, and ∞-vector norms it can be shown that

$$\begin{aligned}
\|A\|_1 &= \max_{1\le j\le n}\sum_{i=1}^{m}|a_{ij}|, &&\text{“max column sum”},\\
\|A\|_\infty &= \max_{1\le i\le m}\sum_{j=1}^{n}|a_{ij}|, &&\text{“max row sum”},\\
\|A\|_2 &= \left(\rho(A^*A)\right)^{1/2} = \sigma_{\max}(A), &&\text{spectral norm},
\end{aligned}$$

where the spectral radius

$$\rho(B) = \max\{\,|\lambda| : \det(B-\lambda I) = 0\,\},$$

and where $\sigma_{\max}(A)$ denotes the largest singular value of A. To remember the formulae for the 1- and ∞-norms, note that 1 is a vertical symbol (for columns) and ∞ is a horizontal symbol (for rows).

A norm is *consistent* if it satisfies $\|AB\| \le \|A\|\,\|B\|$ whenever the product AB is defined. The Frobenius norm and all subordinate norms are consistent. An example of a norm that is not consistent is the “max norm” $\|A\| = \max_{i,j}|a_{ij}|$. The best bound that holds for all $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times p}$ is $\|AB\| \le n\|A\|\,\|B\|$, with equality when $a_{ij} \equiv 1$ and $b_{ij} \equiv 1$.

Table 6.1. *Constants apq such that* $\|x\|_p \le apq\|x\|_q$, $x \in \mathbb{C}^n$.

			q	
		1	2	∞
p	1	1	$\sqrt{n}$	n
	2	1	1	$\sqrt{n}$
	∞	1	1	1

A norm for which $\|UAV\| = \|A\|$ for all unitary U and V is called a *unitarily invariant norm.* These norms have an interesting theory, which we will not explore here (see [581, 1991, §3.5] or [954, 1990, §2.3]). Only two unitarily invariant norms will be needed for our analysis: the 2-norm and the Frobenius norm. That these two norms are unitarily invariant follows easily from the formulae above. For any unitarily invariant norm, the useful property holds that $\|A^*\| = \|A\|$. The 2-norm satisfies the additional relation $\|A^*A\|_2 = \|A\|_2^2$.

The unitary invariance of the 2- and Frobenius norms has implications for error analysis, for it means that multiplication by unitary matrices does not magnify errors. For example, if $A \in \mathbb{C}^{n\times n}$ is contaminated by errors E and Q is unitary, then

$$Q(A+E)Q^* = QAQ^*+F,$$

and $\|F\|_2 = \|QEQ^*\|_2 = \|E\|_2$. In contrast, if we do a general, nonsingular similarity transformation

$$X(A+E)X^{-1} = XAX^{-1}+G,$$

then $\|G\|_2 = \|XEX^{-1}\|_2 \le \kappa_2(X)\|E\|_2$, where $\kappa(X) = \|X\|\;\|X^{-1}\|$ is the *condition number* of X. The condition number satisfies $\kappa(X) \ge 1$ ($\kappa_F(X) \ge \sqrt{n}$) and can be arbitrarily large.

In perturbation theory and error analysis it is often necessary to switch between norms. Therefore inequalities that bound one norm in terms of another are required. It is well known that on a finite-dimensional space any two norms differ by at most a constant that depends only on the dimension (so-called norm equivalence). Tables 6.1 and 6.2 give attainable inequalities for the vector and matrix norms of most interest.

The definition of subordinate matrix norm can be generalized by permitting different norms on the input and output space:

$$\|A\|_{\alpha,\beta} = \max_{x\neq 0} \frac{\|Ax\|_\beta}{\|x\|_\alpha}. \tag{6.6}$$

Table 6.2. *Constants* a_{pq} *such that* $\|A\|_p \le a_{pg}\|A\|_q$, $A \in \mathbb{C}^{m\times n}$. *Here,* $\|A\|_M :=$ *max*$_{i,j}$ $|a_{ij}|$ *and* $\|A\|_s := \sum_{i,j} |a_{ij}|$.

		q: 1	2	∞	F	M	S
p	1	1	$\sqrt{m}$	m	$\sqrt{m}$	m	1
	2	$\sqrt{n}$	1	$\sqrt{m}$	1	$\sqrt{mn}$	1
	∞	n	$\sqrt{n}$	1	$\sqrt{n}$	n	1
	F	$\sqrt{n}$	$\sqrt{\operatorname{rank}(A)}$	$\sqrt{m}$	1	$\sqrt{mn}$	1
	M	1	1	1	1	1	1
	S	n	$\sqrt{mn\operatorname{rank}(A)}$	m	$\sqrt{mn}$	mn	1

Note that, in general, the submultiplicative property $\|AB\|_{a,b} \le \|A\|_{a,b}\|B\|_{a,b}$ does not hold. but we do have

$$\|AB\|_{\alpha,\beta} \le \|A\|_{\gamma,\beta}\|B\|_{\alpha,\gamma}, \tag{6.7}$$

for any third vector norm $\|\cdot\|_\gamma$. The choice $a = 1$ and $b = \infty$ produces the max norm, mentioned above, $\|A\|_{1,\infty} = \max_{i,j} |a_{ij}|$.

At least two important results about matrix norms hold for this mixed subordinate norm. The first is a simple formula for the matrix condition number of a nonsingular $A \in \mathbb{C}^{n\times n}$, defined by

$$\kappa_{\alpha,\beta}(A) := \lim_{\epsilon\to 0} \sup_{\|\Delta A\|_{\alpha,\beta}\le\epsilon\|A\|_{\alpha,\beta}} \left(\frac{\|(A+\Delta A)^{-1} - A^{-1}\|_{\beta,\alpha}}{\epsilon\|A^{-1}\|_{\beta,\alpha}} \right).$$

Note that this definition uses the $\|\cdot\|_{a,b}$ norm on the data space and the $\|\cdot\|_{b,a}$ norm on the solution space, as is natural.

We need the following lemma.

Lemma 6.3. *Given vector norms* $\|\cdot\|_a$ *and* $\|\cdot\|_b$ *and vectors* $x, y \in \mathbb{C}^n$ *such that* $\|x\|_a = \|y\|_b = 1$, *there exists a matrix* B *with* $\|B\|_{a,b} = 1$ *such that* $Bx = y$.

Proof. Recall that the dual of the a-norm is defined by $\|z\|_\alpha^D = \max_{\|w\|_a=1}|z^*w|$. Let z be a vector dual to x, so that $z^*x = \|z\|_\alpha^D\|x\|_\alpha = 1$, and hence $\|z\|_\alpha^D = 1$. Let $B = yz^*$. Then $Bx = y$ and

$$\|B\|_{\alpha,\beta} = \max_{\|w\|_\alpha=1} \|yz^*w\|_\beta = \|y\|_\beta \max_{\|w\|_\alpha=1} |z^*w| = \|y\|_\beta\|z\|_\alpha^D = 1,$$

as required. $\square$

Theorem 6.4. *For nonsingular* $A \in \mathbb{C}^{n\times n}$, *the matrix condition number* $\kappa_{a,b}(A)$ *satisfies*

$$\kappa_{a,b}(A) = \|A\|_{a,b}\|A^{-1}\|_{b,a}. \tag{6.8}$$

Proof. In view of the expansion

$$(A+\Delta A)^{-1} - A^{-1} = -A^{-1}\Delta A A^{-1} + O(\|\Delta A\|^2),$$

the result is proved if we can show that

$$\sup_{\|\Delta A\|_{\alpha,\beta}\le 1} \|A^{-1}\Delta A A^{-1}\|_{\beta,\alpha} = \|A^{-1}\|_{\beta,\alpha}^2. \tag{6.9}$$

That (6.9) holds with the equality replaced by "$\le$" follows from two applications of (6.7). To show the opposite inequality, we have

$$\|A^{-1}\Delta A A^{-1}\|_{\beta,\alpha} = \max_{\|y\|_\beta=1} \|A^{-1}\Delta A A^{-1}y\|_\alpha \ge \|A^{-1}\Delta A x\|_\alpha \|A^{-1}\|_{\beta,\alpha}, \tag{6.10}$$

where, for the lower bound, we have chosen y so that $\|A^{-1}y\|_a = \|A^{-1}\|_{b,a}$, and where $A^{-1}y = \|A^{-1}\|_{b,a}x$ with $\|x\|_a = 1$. Now, from Lemma 6.3, there exists a matrix AA with $\|\Delta A\|_{a,b} = 1$ such that $\Delta A x = y$. In (6.10) this gives $\|A^{-1}\Delta A A^{-1}\|_{\beta,\alpha} \ge \|A^{-1}\|_{\beta,\alpha}^2$, as required. □

The next result concerns the relative distance to singularity for a matrix $A \in \mathbb{C}^{n\times n}$:

$$\mathrm{dist}_{\alpha,\beta}(A) := \min\left\{ \frac{\|\Delta A\|_{\alpha,\beta}}{\|A\|_{\alpha,\beta}} : A + \Delta A \text{ singular} \right\}.$$

It states that the relative distance to singularity is the reciprocal of the condition number.

Theorem 6.5 (Gastinel, Kahan). *For nonsingular* $A \in \mathbb{C}^{n\times n}$, *we have*

$$\mathrm{dist}_{a,b}(A) = (\|A\|_{a,b}\|A^{-1}\|_{b,a})\text{-}1 = \kappa_{a,b}(A)^{-1}.$$

Proof. If $A + \Delta A$ is singular, then there exists $x \ne 0$ such that $(A + \Delta A)x = 0$. Hence

$$\|x\|_\alpha = \|A^{-1}\Delta A x\|_\alpha \le \|A^{-1}\|_{\beta,\alpha}\|\Delta A x\|_\beta \le \|A^{-1}\|_{\beta,\alpha}\|\Delta A\|_{\alpha,\beta}\|x\|_\alpha,$$

giving

$$\frac{\|\Delta A\|_{\alpha,\beta}}{\|A\|_{\alpha,\beta}} \ge \kappa_{\alpha,\beta}(A)^{-1}.$$

To show that a suitable perturbation achieves this lower bound, let y be such that $\|y\|_b = 1$ and $\|A^{-1}y\|_a = \|A^{-1}\|_{b,a}$, and write $x = A^{-1}y$. By Lemma 6.3 there exists B with $\|B\|_{a,b} = 1$ such that $Bx/\|x\|_a = -y$. Letting $\Delta A = B/\|x\|_a$ we have $\|\Delta A\|_{a,b}/\|A\|_{a,b} = \kappa_{a,b}(A)^{-1}$. and $A+\Delta A$ is singular because $(A + \Delta A)A^{-1}y = 0$. □

6.3. The Matrix *p*-Norm

The matrix p-norm is the norm subordinate to the Hölder p-norm:

$$\|A\|_p = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}, \qquad p \geq 1. \tag{6.11}$$

Formulae for $\|A\|_p$ are known only for $p = 1, 2, \infty$. For other values of p, how to estimate or compute $\|A\|_p$ is an interesting problem, the study of which, as well as being interesting in its own right, yields insight into the properties of the 1, 2, and ∞ norms.

By taking $x = e_j$ in (6.11). using (6.4). and using (6.21) below, we can derive the bounds, for $A \in \mathbb{C}^{m\times n}$,

$$\max_j \|A(:,j)\|_p \leq \|A\|_p \leq n^{1-1/p} \max_j \|A(:,j)\|_p, \tag{6.12}$$

$$\max_i \|A(i,:)\|_{p/(p-1)} \leq \|A\|_p \leq m^{1/p} \max_i \|A(i,:)\|_{p/(p-1)}. \tag{6.13}$$

Matrix norms can be compared using the following elegant result of Schneider and Strang [900, 1962] (see also [580, 1985, Thm. 5.6.18]): if $\|\cdot\|_a$ and $\|\cdot\|_b$ denote two vector norms and the corresponding subordinate matrix norms, then for $A \in \mathbb{C}^{m\times n}$

$$\max_{A\neq 0} \frac{\|A\|_\alpha}{\|A\|_\beta} = \left(\max_{0\neq x\in\mathbb{C}^m} \frac{\|x\|_\alpha}{\|x\|_\beta} \right) \left(\max_{0\neq x\in\mathbb{C}^n} \frac{\|x\|_\beta}{\|x\|_\alpha} \right). \tag{6.14}$$

From (6.4) and (6.14), we have, when $m = n$,

$$\max_{A\neq 0} \frac{\|A\|_{p_1}}{\|A\|_{p_2}} = n^{\left(\frac{1}{\min(p_1,p_2)} - \frac{1}{\max(p_1,p_2)}\right)}. \tag{6.15}$$

Note that, unlike for vectors, $p_1 < p_2$ does not imply $\|A\|_{p_1} \geq \|A\|_{p_2}$. The result (6.15) implies, for example, that for all $p \geq 1$

$$\frac{\|A\|_1}{n^{1-1/p}} \leq \|A\|_p \leq n^{1-1/p}\|A\|_1, \tag{6.16}$$

$$\frac{\|A\|_2}{n^{|1/p-1/2|}} \leq \|A\|_p \leq n^{|1/p-1/2|}\|A\|_2. \tag{6.17}$$

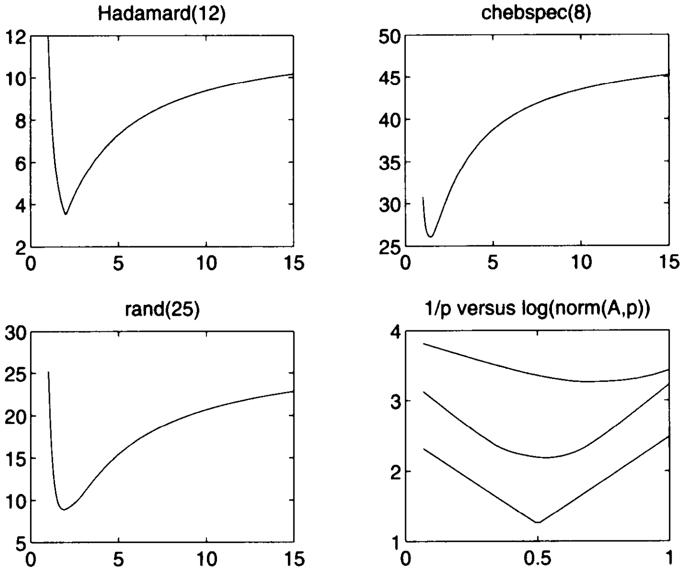


Figure 6.1. *Plots of* p *versus* $\|A\|_p$, *for* $1 \le p \le 15$. *Fourth plot shows* $1/p$ *versus* $\log \|A\|_p$ *for the matrices in the first three plots.*

Upper bounds for $\|A\|_p$ that do not involve m or n can be obtained from the interesting property that $\log \|A\|_p$ is a convex function of $1/p$ for $p \ge 1$ (see Figure 6.1), which is a consequence of the Riesz-Thorin theorem [503, 1952, pp. 214, 219], [450, 1991]. The convexity implies that if $f(\alpha) = \|A\|_{1/\alpha}$, then for $0 \le \alpha, \beta \le 1$,

$$\log f(\theta\alpha + (1-\theta)\beta) \le \theta \log f(\alpha) + (1-\theta)\log f(\beta), \qquad 0 \le \theta \le 1.$$

Writing $p_1 = 1/\alpha$ and $p_2 = 1/\beta$, this inequality can be expressed as

$$\|A\|_p \le \|A\|_{p_1}^{\theta} \|A\|_{p_2}^{1-\theta}, \qquad p = \frac{p_1 p_2}{(1-\theta)p_1 + \theta p_2}, \tag{6.18}$$

$$1 \le p_1, p_2 \le \infty, \quad 0 \le \theta \le 1.$$

Two interesting special cases are

$$\|A\|_p \le \|A\|_1^{1/p} \|A\|_\infty^{1-1/p}, \tag{6.19}$$

and

$$\|A\|_p \le \|A\|_1^{2/p-1} \|A\|_2^{2-2/p}, \qquad 1 \le p \le 2. \tag{6.20}$$

Note that (6.19) includes the well-known inequality $\|A\|_2 \le \sqrt{\|A\|_1 \|A\|_\infty}$.

Two further results that are familiar for $p = 1, 2, \infty$ are

$$\|A^*\|_p = \|A\|_q, \qquad \frac{1}{p} + \frac{1}{q} = 1, \tag{6.21}$$

(see, for example, [580, 1985, Thm. 5.6.36]) and

$$\left\| \begin{bmatrix} 0 & A \\ A^* & 0 \end{bmatrix} \right\|_p = \max(\|A\|_p, \|A\|_q).$$

The bounds (6.16) and (6.17) imply that given the ability to compute $\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$ we can estimate $\|A\|_p$ correct to within a factor $n^{1/4}$. These a priori estimates are at their best when p is close to 1, 2, or ∞, but in general they will not provide even one correct significant digit. The bound in (6.18) can be much smaller than the other upper bounds given above, but how tight it is depends on how nearly $\log \|A\|_p$ is linear in p. Numerical methods are needed to obtain better estimates: these are developed in chapter 14.

6.4. Notes and References

The matrix condition number appears to have been first introduced explicitly by Turing [1027, 1948], who defined, for example, the N-condition number of $A \in \mathrm{IR}^{n\times x}$ as $n^{-1} N(A)N(A^{-1})$, where $N(\cdot)$ is Turing's notation for the Frobenius norm. Todd [1003, 1968] gives a short survey of the matrix condition number with many references.

Theorem 6.2 was originally proved by Bauer, Stoer, and Witzgall, in a paper that contains many interesting results on monotonic norms [84, 1961].

Tables of constants in inequalities between different norms have been given by various authors: see, for example, Stone [957, 1962] and Zielke [1129, 1988].

Our development of the mixed subordinate norm $\|\cdot\|_{a,b}$ is based on that of D. J. Higham [526, 1995].

Theorem 6.5 is proved by Kahan [626, 1966, pp. 775 776], who attributes it to Gastinel but gives no reference. For the 2-norm, this result goes back to a paper by Eckart and Young [334, 1936]. Theorem 6.5 is an instance of a relationship that holds for many problems: the condition number is the reciprocal of the distance to the nearest singular problem (one with an infinite condition number). This relationship applies to matrix inversion, eigenvalue and eigenvector computation, polynomial zero-finding, and pole assignment in linear control systems. For an in-depth study see Demmel [281, 1987].

Direct proofs of inequality (6.19) can be found in Kato [646, 1976, p. 29] and Todd [1006, 1977, pp. 25-26]. The inequality does not seem to be well known.

For historical comments on the development of norms in numerical analysis, see Householder [587, 1964, Chap. 2] and Stewart and Sun [954, 1990, Chap. 2].

Problems

> *Problems worthy*
> *of attack*
> *prove their worth*
> *by hitting back.*
>
> -PIET HEIN, *Grooks (1966)*

6.1. Prove the inequalities given in Tables 6.1 and 6.2. Show that each inequality in Table 6.2 (except the one for $a_{S,2}$) is attainable for a matrix of the form $A = xy^T$, where $x, y \in \{e, e_j\}$, where $e = [1, 1, \ldots, 1]^T$. Show that equality in $\|A\|_S \le a_{s,2} \|A\|_2$ is attained for square real matrices A iff A is a scalar multiple of a Hadamard matrix (see §9.3 for the definition of a Hadamard matrix), and for square complex matrices if $a_{rs} = \exp(2\pi i(r - 1)(s - 1)/n)$ (this is a Vandermonde matrix based on the roots of unity).

6.2. Let $x, y \in \mathbb{C}^{n\times n}$. Show that, for any subordinate matrix norm, $\|xy^*\| = \|x\| \, \|y\|_D$.

6.3. Show that a subordinate matrix norm $\|\cdot\|$ on $\mathbb{C}^{n\times n}$ satisfies

$$\|A\| = \max_{x,y\neq 0} \frac{\operatorname{Re} y^* A x}{\|y\|_D \|x\|}.$$

> *From ancient times until now the*
> *study of magic squares has flourished as a kind of cult,*
> *often with occult trappings, whose initiates range from*
> *such eminent mathematicians as Arthur Cayley and Oswald Veblen*
> *to laymen such as Benjamin Franklin.*
>
> -MARTIN GARDNER, *More Mathematical Puzzles and Diversions* (1961)

6.4. Let $M_n \in \mathbb{R}^{n\times n}$ denote a magic square matrix, that is, an $n \times n$ matrix containing the integers from 1 to n^2 arranged in such a way that the row and column sums are all the same. Let μ_n, denote the magic sum of M_n (thus, $\mu_n = n(n^2 + 1)/2$). Show that $\|M_n\|_p = \mu_n$ for all $1 \le p \le \infty$. (This result is a special case of an apparently little-known result of Stoer and Witzgall, which states that the norm of a doubly stochastic matrix is 1 for any norm subordinate to a permutation-invariant absolute vector norm [956, 1962].)

6.5. Show that $\|ABC\|_F \le \|A\|_2 \|B\|_F \|C\|_2$ for any A, B, and C such that the product is defined. (This result remains true when the Frobenius norm is replaced by any unitarily invariant norm [581, 1991, p. 211].)

6.6. Show that for any nonsingular $A \in \mathbb{C}^{n\times n}$,

$$\kappa_{\alpha,\beta}(A) = \frac{\max_{x\neq 0} \dfrac{\|Ax\|_\beta}{\|x\|_\alpha}}{\min_{x\neq 0} \dfrac{\|Ax\|_\beta}{\|x\|_\alpha}}.$$

6.7. Show that for any $A \in \mathbb{C}^{n\times n}$ and any consistent matrix norm, $\rho(A) \le \|A\|$, where ρ is the spectral radius.

6.8. Show that for any $A \in \mathbb{C}^{n\times n}$ and $\delta > 0$ there is a consistent norm $\|\cdot\|$ (which depends on A and δ) such that $\|A\| \le \rho(A) + \delta$, where ρ is the spectral radius. Hence show that if $\rho(A) < 1$ then there is a consistent norm $\|\cdot\|$ such that $\|A\| < 1$.

6.9. Let $A \in \mathbb{C}^{m\times n}$. Use the SVD to find expressions for $\|A\|_2$ and $\|A\|_F$ in terms of the singular values of A. Hence obtain a bound of the form $c_1\|A\|_2 \le \|A\|_F \le c_2\|A\|_2$, where c_1 and c_2 are constants that depend on n. When is there equality in the upper bound? When is there equality in the lower bound?

6.10. Show that

$$\left\| \begin{bmatrix} I & F \\ 0 & I \end{bmatrix} \right\|_2 = \sqrt{\frac{2 + \|F\|_2^2 + \|F\|_2\sqrt{4 + \|F\|_2^2}}{2}}.$$

Deduce that when $\|F\|_2 = 1$, the norm is $(1+\sqrt{5})/2$, the golden ratio.

6.11. Let $A \in \mathbb{C}^{n\times n}$. Prove that (a) $\|A\|_{1,b} = \max_j \|A(:,j)\|_b$, and (b) $\|A\|_{a,\infty} = \max_i \|A(i,:)\|_{\alpha}^{D}$. What is $\|A\|_{1,\infty}$?

6.12. (Tao [994, 1984]) Show that if A is Hermitian positive definite then

$$\|A\|_{\infty,1} = \max\{\, x^*Ax : \|x\|_\infty = 1 \,\}.$$

(Rohn [879, 1995] shows that the problem of computing $\|A\|_{\infty,1}$ is NP-hard.)

6.13. Prove that if $H \in \mathbb{R}^{n\times n}$ is a Hadamard matrix then

$$\|H\|_p = \max\{n^{1/p}, n^{1-1/p}\}.$$

(See §9.3 for the definition of a Hadamard matrix.)

6.14. Show that if $A \in \mathbb{R}^{m\times n}$ has at most μ nonzeros per row then

$$\max_j \|A(:,j)\|_p \le \|A\|_p \le \mu^{1-1/p} \max_j \|A(:,j)\|_p, \qquad (6.22)$$

while if A has at most m nonzeros per column then

$$\max_i \|A(i,:)\|_q \le \|A\|_p \le \mu^{1/p} \max_i \|A(i,:)\|_q, \tag{6.23}$$

where $p^{-1} + q^{-1} = 1$. (These inequalities generalize (6.12) and (6.13).)

6.15. Show that if $A \in \mathbb{C}^{n\times n}$ then for any p-norm ($1 \le p \le \infty$),

$$\|A\|_p \le \|\,|A|\,\|_p \le n^{\min(1/p,1-1/p)}\|A\|_p \le \sqrt{n}\|A\|_p.$$

6.16. Define the function $\nu : \mathbb{C}^n \to \mathbb{R}$ by

$$\nu(x) = \sum_{i=1}^{n} \big(|\operatorname{Re} x_i| + |\operatorname{Im} x_i|\big).$$

Is ν a vector norm on $\mathbb{C}^n$? Derive an explicit expression for

$$\nu(A) = \max_{\nu(x)=1} \nu(Ax), \quad A \in \mathbb{C}^{n\times n}.$$

Chapter 7

Perturbation Theory for Linear Systems

Our hero is the intrepid, yet sensitive matrix A.
Our villain is E, who keeps perturbing A.
When A is perturbed he puts on a crumpled hat: $\tilde{A} = A + E$.

-G. W. STEWART and JI-GUANG SUN, *Matrix Perturbation Theory (1990)*

The expression 'ill-conditioned' is sometimes used merely as a
term of abuse applicable to matrices or equations . . .
It is characteristic of ill-conditioned sets of equations that
small percentage errors in the coefficients given may lead to
large percentage errors in the solution.

-A. M. TURING, *Rounding-Off Errors in Matrix Processes* (1948)

In this chapter we are concerned with a linear system $Ax = b$, where $A \in \mathrm{IR}^{n \times n}$. In the context of uncertain data or inexact arithmetic there are three important questions:

(1) How much does x change if we perturb A and b; that is, how sensitive is the solution to perturbations in the data?

(2) How much do we have to perturb the data A and b for an approximate solution y to be the exact solution of the perturbed system-in other words, what is the backward error of y?

(3) What bound should we compute in practice for the forward error of a given approximate solution?

To answer these questions we need both normwise and componentwise perturbation theory.

7.1. Normwise Analysis

First, we present some classical normwise perturbation results. We denote by $\|\cdot\|$ any vector norm and the corresponding subordinate matrix norm. As usual, $\kappa(A) = \|A\| \, \|A^{-1}\|$ is the matrix condition number. Throughout this chapter the matrix E and the vector f are arbitrary and represent tolerances against which the perturbations are measured (their role becomes clear when we consider componentwise results).

Our first result makes precise the intuitive feeling that if the residual is small then then have a "good" approximate solution.

Theorem 7.1 (Rigal and Gaches). The normwise backward error

$$h_{E,f}(y) := \min\{ \epsilon : (A + \Delta A)_y = b + \Delta b, \ \|\Delta A\| \le \epsilon\|E\|, \ \|\Delta b\| \le \epsilon\|f\|\} \tag{7-1}$$

is given by

$$\eta_{E,f}(y) = \frac{\|r\|}{\|E\| \, \|y\| + \|f\|}, \tag{7.2}$$

where $r = b - Ay$.

Proof. It is straight forward to show that the right-hand side of (7.2) is a lower bound for $h_{E,f}(y)$. This lower bound is attained for the perturbations

$$\Delta A_{\min} = \frac{\|E\| \, \|y\|}{\|E\| \, \|y\| + \|f\|} r z^T, \qquad \Delta b_{\min} = -\frac{\|f\|}{\|E\| \, \|y\| + \|f\|} r, \tag{7.3}$$

where z is a vector dual to y (see §6.1). □

For the particular choice $E = A$ and $f = b$, $h_{E,f}(y)$ is called the *normwise relative backward error.*

The next result measures the sensitivity of the system.

Theorem 7.2. *Let* $Ax = b$ *and* $(A + \Delta A)y = b + \Delta b$, *where* $\|\Delta A\| \le \epsilon\|E\|$ and $\|\Delta b\| \le \epsilon\|f\|$, *and assume that* $\epsilon\|A^{-1}\|\|E\| < 1$. *Then*

$$\frac{\|x-y\|}{\|x\|} \le \frac{\epsilon}{1-\epsilon\|A^{-1}\|\,\|E\|}\left(\frac{\|A^{-1}\|\,\|f\|}{\|x\|} + \|A^{-1}\|\,\|E\|\right), \qquad (7.4)$$

and this bound is attainable to first order in ϵ.

Proof. The bound (7.4) follows easily from the equation $A(y - x) = \Delta b - \Delta Ax + \Delta A(x - y)$. It is attained to first order in ϵ for $DA = \epsilon\|E\|\|x\|wv^T$ and $\Delta b = -\epsilon\|f\|w$, where $\|w\| = 1$, $\|A^{-1}w\| = \|A^{-1}\|$ and v is a vector dual to x. □

Associated with the way of measuring perturbations used in these two theorems is the normwise condition number

$$\kappa_{E,f}(A,x) := \lim_{\epsilon\to 0}\sup\Big\{\frac{\|\Delta x\|}{\epsilon\|x\|} : (A+\Delta A)(x+\Delta x) = b+\Delta b,$$
$$\|\Delta A\| \le \epsilon\|E\|,\ \ \|\Delta b\| \le \epsilon\|f\|\Big\}.$$

Because the bound of Theorem 7.2 is sharp, it follows that

$$\kappa_{E,f}(A,x) = \frac{\|A^{-1}\|\,\|f\|}{\|x\|} + \|A^{-1}\|\,\|E\|.$$

For the choice $E = A$ and $f = b$ we have $\kappa(A) \le \kappa_{E,f}(A, x) \le 2\kappa(A)$, and the bound (7.4) can be weakened slightly to yield the familiar form

$$\frac{\|x-y\|}{\|x\|} \le \frac{2\epsilon\kappa(A)}{1-\epsilon\kappa(A)}.$$

A numerical example illustrates the above results. Let A be the 8×8 Vandermonde matrix with (i,j) element $j^{2(i-1)}$, and let $b = e_1$ be the first unit vector, so that x is the first column of A^{-1}. We take y to be the approximate solution to $Ax = b$ computed by Gaussian elimination with partial pivoting. Computations are performed in MATLAB ($u \approx 1.1 \times 10^{-16}$). We find that $h_{A,b}(y) = 3.05 \times 10^{-21}$ for the ∞-norm, and $\kappa_\infty(A) = 1.68 \times 10^{13}$. This is an admirably small backward error, but it may be uninformative for two reasons. First, the elements of A vary over 12 orders of magnitude, so while our backward error perturbations are small compared with the large elements of A, we may be making large perturbations in the small elements (indeed we *are* in this particular example). Second, we are perturbing the zero elements of b (as can be seen from (7.3) together with the fact that for this example the residual r has no zero entries); this is unsatisfactory if we wish to regard y as the first column of the inverse of a perturbed matrix.

Next. let $b = Ae$. where $e = [1, 1,. . . . 1]^T$, and let z be the solution to the perturbed q-stem $(A + \Delta A)z = b + \Delta b$, where $\Delta A = \text{tol}|A|$ and $\Delta b = \text{tol}|b|$, with to1 = $8u$. We find that

$$\frac{\|x-z\|_\infty}{\|x\|_\infty} = 2.40 \times 10^{-12}, \tag{7.5}$$

while the corresponding bound from (7.4) with ϵ = tol, $E = A$, and $f = b$ is 3.03×10^{-2}. Thus the normwise forward error bound is extremely pessimistic for this special choice of perturbation.

To obtain a more satisfactory backward error measure and a sharper perturbation bound in this example, we need componentwise analysis.

7.2. Componentwise Analysis

The componentwise backward error is defined as

$$\omega_{E,f}(y) = \min\{\epsilon : (A + \Delta A)y = b + \Delta b,\ \ |\Delta A| \le \epsilon E,\ \ |\Delta b| \le \epsilon f\}, \tag{7.6}$$

where E and f are now assumed to have nonnegative entries. Inequalities between matrices or vectors are understood to hold componentwise. In this definition each element of a perturbation is measured relative to its individual tolerance, so, unlike in the normwise definition, we are making full use of the $n^2 + n$ parameters in E and f.

How should E and f be chosen? The most common choice of tolerances is $E = |A|$ and $f = |b|$, which yields the *componentwise relative backward error.* For this choice

$$a_{ij} = 0 \ \Rightarrow\ \Delta a_{ij} = 0 \quad \text{and} \quad b_i = 0 \ \Rightarrow\ \Delta b_i = 0$$

in (7.6), and so if $w_{E,f}(y)$ is small then y solves a problem that is close to the original one in the' sense of componentwise relative perturbations and has the same sparsity pattern. Another attractive property of the componentwise relative backward error is that it is insensitive to the scaling of the system: if $Ax = b$ is scaled to $(S_1AS_2)(S_2^{-1}x) = S_1b$, where S_1 and S_2 are diagonal, and y is scaled to $S_2^{-1}y$, then w remains unchanged.

The choice $E = |A|ee^T$, $f = |b|$ gives a row-wise backward error. The constraint $|\Delta A| \le \epsilon E$ is now $|\Delta a_{ij}| \le \epsilon\alpha_i$, 'where $(a_i$ is the 1-norm of the ith row of A, so perturbations to the ith row of A are being measured relative to the norm of that row. A columnwise backward error can be formulated in a similar way, by taking $E = ee^T|A|$ and $f = \|b\|_\infty e$.

The third natural choice of tolerances is $E = \|A\|ee^T$ and $f = \|b\|e$, for which $w_{E,f}(y)$ is the same as the normwise backward error $h_{E,f}(y)$. up to a constant.

As for the normwise backward error in Theorem 7.1, there is a simple formula for $w_{E,f}(g)$.

Theorem 7.3 (Oettli and Prager). *The componentwise backward error is given by*

$$\omega_{E,f}(y) = \max_i \frac{|r_i|}{(E|y| + f)_i}, \tag{7.7}$$

where $r = b - Ay$, and $\xi/0$ is interpreted as zero if $\xi = 0$ and infinity otherwise.

Proof. It is easy to show that the right-hand side of (7.7) is a lower bound for $w(y)$, and that this bound is attained for the perturbations

$$\Delta A = D_1 E D_2, \qquad \Delta b = -D_1 f, \tag{7.8}$$

where $D_1 = \mathrm{diag}(r_i/(E|y| + f)_i)$ and $D_2 = \mathrm{diag}(\mathrm{sign}(y_i))$. □

The next result gives a forward error bound corresponding to the componentwise backward error.

Theorem 7.4. *Let $Ax = b$ and $(A + \Delta A)y = b + \Delta b$, where $|\Delta A| \le \epsilon E$ and $|\Delta b| \le \epsilon f$, and assume that $\epsilon \| \, |A^{-1}|E \, \| < 1$, where $\|\cdot\|$ is an absolute norm. Then*

$$\frac{\|x - y\|}{\|x\|} \le \frac{\epsilon}{1 - \epsilon \| \, |A^{-1}|E \, \|} \frac{\| \, |A^{-1}|E|x| + |A^{-1}|f \, \|}{\|x\|}, \tag{7.9}$$

and for the ∞-norm this bound is attainable to first order in ϵ.

Proof. The bound (7.9) follows easily from the equation $A(y - x) = \Delta b - \Delta A x + \Delta A(x - y)$. For the ∞-norm the bound is attained, to first order in ϵ, for $\Delta A = \epsilon D_1 E D_2$ and $\Delta b = -\epsilon D_1 f$, where $D_2 = \mathrm{diag}(\mathrm{sign}(x_i))$ and $D_1 = \mathrm{diag}(\xi_j)$, where $\xi_j = \mathrm{sign}(A^{-1})_{kj}$ and $\| \, |A^{-1}|E|x| + |A^{-1}|f \, \|_\infty = \left(|A^{-1}|E|x| + |A^{-1}|f\right)_k$. □

Theorem 7.4 implies that the condition number

$$\mathrm{cond}_{E,f}(A,x) := \lim_{\epsilon \to 0} \sup \Big\{ \frac{\|\Delta x\|_\infty}{\epsilon \|x\|_\infty} : (A + \Delta A)(x + \Delta x) = b + \Delta b, \\ |\Delta A| \le \epsilon E, \ |\Delta b| \le \epsilon f \Big\}$$

is given by

$$\mathrm{cond}_{E,f}(A,x) = \frac{\| \, |A^{-1}|E|x| + |A^{-1}|f \, \|_\infty}{\|x\|_\infty}. \tag{7.10}$$

For the special case $E = |A|$ and $f = |b|$ we have the condition numbers introduced by Skeel [919, 1979]:

$$\mathrm{cond}(A,x) := \frac{\| \, |A^{-1}||A||x| \, \|_\infty}{\|x\|_\infty},$$

which differs from $\operatorname{cond}_{|A|,|b|}(A, x)$ by at most a factor 2. and

$$\operatorname{cond}(A) := \operatorname{cond}(A, e) = \| \, |A^{-1}||A| \, \|_\infty \le \kappa_\infty(A). \tag{7.11}$$

How does cond compare with κ? Since cond(A) is invariant under row scaling $Ax = b \longrightarrow (DA)x = Db$, where D is diagonal. it can be arbitrarily smaller than $\kappa_\infty(A)$. In fact, it is straightforward to show that

$$\min\{ \kappa_\infty(DA) : D \text{ diagonal} \} = \operatorname{cond}(A), \tag{7.12}$$

where the optimal scaling D_R equilibrates the rows of A, that is, D_RA has rows of unit 1-norm ($D_R|A|e=e$)

Chandrasekaran and Ipsen [197, 1995] note the following inequalities. First. with D_R as just defined,

$$\frac{\kappa_\infty(A)}{\kappa_\infty(D_R)} \le \operatorname{cond}(A) \le \kappa_\infty(A) \tag{7.13}$$

(these inequalities imply (7.12)). Thus cond(A) can be much smaller than $\kappa_\infty(A)$ only when the rows of A are badly scaled. Second, if D_C, equilibrates the columns of A ($e^T|A|D_C=e^T$) then

$$\frac{\kappa_1(A)}{n\kappa_\infty(D_C)} \min_j \frac{\|A^{-1}e_j\|_\infty}{\|A^{-1}\|_1} \le \operatorname{cond}(A, x) \le \kappa_\infty(A).$$

These inequalities show that cond(A, x) can be much smaller than $\kappa_\infty(A)$ only when the columns of either A or A^{-1} are badly scaled.

Returning to the numerical example of §7.1, we find that $w_{E,f}(y) = 1.10 \times 10^{-12}$ for $E = |A|$ and $f = |b|$ or $f = 0$. This tells us that if we measure changes to A in a componentwise relative sense, then for y to be the first column of the inverse of a perturbed matrix we must make relative changes to A four orders of magnitude larger than the unit roundoff. For the perturbed system. Theorem 7.4 with ϵ = tol, $E = |A|$, and $f = |b|$ gives the bound

$$\frac{\|x - z\|_\infty}{\|x\|_\infty} \le 4.08 \times 10^{-10},$$

which is eight orders of magnitude smaller than the normwise bound from Theorem 7.2, and only a factor 170 larger than the actual forward error (7.5).

An example of Kahan [626, 1966] is also instructive Let

$$A = \begin{bmatrix} 2 & -1 & 1 \\ -1 & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon \end{bmatrix}, \quad b = \begin{bmatrix} 2(1+\epsilon) \\ -\epsilon \\ \epsilon \end{bmatrix}, \tag{7.14}$$

where $0 < \epsilon \ll 1$, so that $x = [\epsilon, -1, 1]^T$. The normwise condition number $\kappa_\infty(A)$ is $2(1+\epsilon^{-1})$ so the system is very sensitive to arbitrary perturbations in A and b. Moreover,

$$|A^{-1}||A| = \begin{bmatrix} 1 & \epsilon & \epsilon \\ \dfrac{2\epsilon+1}{2\epsilon} & 1 & 1 \\ \dfrac{2\epsilon+1}{2\epsilon} & 1 & 1 \end{bmatrix},$$

so $\mathrm{cond}(A) = 3 + (2\epsilon)^{-1}$, which implies that the system is also very sensitive to componentwise perturbations for some right-hand sides. However, $\mathrm{cond}(A, x) = 5/2 + \epsilon$, so *for this particular* b the system is very well conditioned under componentwise perturbations.

A word is in order concerning the choice of condition number. Every condition number for a linear system is defined with respect to a particular class of perturbations. It is important to use the right condition number for the occasion. For example, if $\widehat{x}$ is a computed solution to $Ax = b$ and we know its normwise backward error $h_{A,b}(\widehat{x})$, then it is the condition number $\kappa(A)$ that appears in the relevant forward error bound (multiplying $\eta_{A,b}(\widehat{x})$) and therefore tells us something about the accuracy of $\widehat{x}$. The componentwise condition number $\mathrm{cond}(A, x)$ is relevant only if we are dealing with the componentwise relative backward error, $w_{|A|,|b|}(\widehat{x})$. Looked at another way, each algorithm has an associated error analysis that determines the condition number relevant to that algorithm.

7.3. Scaling to Minimize the Condition Number

In the last section we noted the invariance of $\mathrm{cond}(A)$ under row scaling, which contrasts with the strong dependence of $\kappa_\infty(A)$ upon the row scaling. The opportunity to scale the rows or columns of A arises in various applications, so we now take a closer look at the effect of scaling on the normwise condition number.

First, we consider one-sided scaling, by giving a generalization of a well-known result of van der Sluis [1039, 1969]. It shows that, for one-sided scaling in a Hölder p-norm, equilibrating the rows or columns is a nearly optimal strategy. We state the result for rectangular matrices A, for which we define $\kappa_p(A) := \|A\|_p \|A^+\|_p$, where A^+ is the pseudo-inverse of A (see Problem 19.3).

Theorem 7.5 (van der Sluis). *Let* $A \in \mathbb{R}^{m\times n}$ *have full rank, let* $\mathcal{D}_k \subset \mathbb{R}^{k\times k}$ *denote the set of nonsingular diagonal matrices, and define*

$$D_C := \mathrm{diag}(\|A(:,j)\|_p)^{-1}, \quad D_R := \mathrm{diag}(\|A(i,:)\|_p)^{-1}.$$

Then

$$\kappa_p(AD_C) \le n^{1-1/p} \min_{D\in\mathcal{D}_n} \kappa_p(AD), \tag{7.15}$$

$$\kappa_p(D_RA) \le m^{1/p} \min_{D\in\mathcal{D}_m} \kappa_p(DA). \tag{7.16}$$

Proof. For any $X \in \mathbb{R}^{m\times n}$ we have, from (6.12),

$$\max_j \|A(:,j)\|_p \le \|A\|_p \le n^{1-1/p} \max_j \|A(:,j)\|_p. \tag{7.17}$$

Therefore

$$\|AD_C\|_p \le n^{1-1/p}. \tag{7.18}$$

Now, for any $D \in \mathcal{D}_k$,

$$\begin{aligned} D_C^{-1}A^+\|_p &= \|D_C^{-1}D\cdot D^{-1}A^+\|_p \\ &\le \max_j\bigl(|d_{jj}|\,\|A(:,j)\|_p\bigr)\|D^{-1}A^+\|_p \\ &\le \|AD\|_p\|D^{-1}A^+\|_p = \kappa_p(AD), \end{aligned} \tag{7.19}$$

using the first inequality in (7.17). Multiplying (7.18) and (7.19) and minimizing over D, we obtain (7.15). Inequality (7.16) follows by noting that $\kappa_p(DA) = \kappa_q(A^TD)$, where $p^{-1} + q^{-1} = 1$ (see (6.21)). □

For $p = \infty$, (7.16) confirms what we already know from (7.12) and (7.13): that in the ∞-norm, row equilibration is an optimal row scaling strategy. Similarly, for $p = 1$, column equilibration is the best column scaling, by (7.15). Theorem 7.5 is usually stated for the 2-norm, for which it shows that row and column equilibration produce condition numbers within factors $\sqrt{m}$ and $\sqrt{n}$, respectively, of the minimum 2-norm condition numbers achievable by row and column scaling.

As a corollary of Theorem 7.5 we have the result that among two-sided diagonal scalings of a symmetric positive definite matrix. the one that gives A a unit diagonal is not far from optimal.

Corollary 7.6 (van der Sluis). *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite and let $D_* = \mathrm{diag}(a_{ii}^{-1/2})$. Then*

$$\kappa_2(D_*AD_*) \le n \min_{D\in\mathcal{D}_n} \kappa_2(DAD). \tag{7.20}$$

Proof. Let $A = R^TR$ be a Cholesky factorization, note that $\kappa_2(DAD) = \kappa_2(RD)^2$, and apply Theorem 7.5 to RD. □

Is the scaling D_* in Corollary 7.6 ever optimal? Forsythe and Straus [386, 1955] show that it is optimal if A is symmetric positive definite with property

A (that is, there exists a permutation matrix P such that PAP^T can be expressed as a block 2×2 matrix whose (1,1) and (2,2) blocks are diagonal). Thus, for example, any symmetric positive definite tridiagonal matrix with unit diagonal is optimally scaled.

We note that by using (6.22) in place of (7.17), the inequalities of Theorem 7.5 and Corollary 7.6 can be strengthened by replacing m and n with the maximum number of nonzeros per column and row, respectively.

Here is an independent result for the Frobenius norm.

Theorem 7.7 (Stewart and Sun). *Let* $A = [a_1, \ldots, a_n] \in \mathbb{R}^{n\times n}$ *be nonsingular, with* $B := A^{-1} = [b_1, \ldots, b_n]^T$, *and let* $D_C = \operatorname{diag}((\|b_j\|_2/\|a_j\|_2)^{1/2})$. *Then*

$$\sum_j \|a_j\|_2\|b_j\|_2 = \kappa_F(AD_C) = \min_{D\in\mathcal{D}_n} \kappa_F(AD).$$

Proof. For $D = \operatorname{diag}(d_i) \in \mathcal{D}_n$, we have, using the Cauchy-Schwarz inequality,

$$\kappa_F(AD) = \Bigl(\sum_j d_j^2\|a_j\|_2^2\Bigr)^{1/2}\Bigl(\sum_j d_j^{-2}\|b_j\|_2^2\Bigr)^{1/2} \ge \sum_j \|a_j\|_2\|b_j\|_2,$$

with equality if $d_j\|a_j\|_2 = \alpha d_j^{-1}\|b_j\|_2$ for all j, for some $a \ne 0$. There is equality for $d_j^2 = \|b_j\|_2/\|a_j\|_2$. ☐

As we have seen in this and the previous section, the minimum value of $\kappa_\infty(DA)$ is $\|\,|A^{-1}||A|\,\|_\infty$. The next result shows that for two-sided scalings the matrix $|A^{-1}||A|$ again features in the formula for the minimal condition number. A matrix is irreducible if it cannot be symmetrically permuted to block triangular form. A Perron vector of $B \ge 0$ is a nonnegative eigenvector corresponding to the eigenvalue $\rho(B)$, where ρ denotes the spectral radius.

Theorem 7.8 (Bauer). *Let* $A \in \mathbb{R}^{n\times n}$ *be nonsingular and suppose that* $|A||A^{-1}|$ *and* $|A^{-1}||A|$ *are irreducible. Then*

$$\min_{D_1,D_2\in\mathcal{D}_n} \kappa_\infty(D_1AD_2) = \rho(|A||A^{-1}|). \tag{7.21}$$

The minimum is attained for $D_1 = \operatorname{diag}(x)^{-1}$ *and* $D_2 = \operatorname{diag}(|A^{-1}|x)$, *where* $x > 0$ *is a right Perron vector of* $|A||A^{-1}|$ (*so that* $|A||A^{-1}| = \rho(|A||A^{-1}|)x$).

Proof. See Problem 7.9. ☐

For the Kahan example (7.14),

$$\rho(|A^{-1}||A|) \approx 2.62 + 1.79\epsilon \ll 3 + (2\epsilon)^{-1} = \|\,|A^{-1}||A|\,\|_\infty,$$

and, in fact, $\kappa_\infty(DAD) = 3$ for $D = \operatorname{diag}(\epsilon^{1/2}, \epsilon^{-1/2}, \epsilon^{-1/2})$, so a symmetric two-sided scaling is nearly optimal in this case.

7.4. The Matrix Inverse

We briefly discuss component wise perturbation theory for the matrix inverse. With $X := A^{-1}$ and $X + \Delta X := (A + \Delta X)^{-1}$, a componentwise condition number is

$$\mu_E(A) := \lim_{\epsilon\to 0} \sup\left\{ \frac{\|\Delta X\|_\infty}{\epsilon\|X\|_\infty} : |\Delta A| \le \epsilon E \right\} \le \frac{\|\,|A^{-1}||E||A^{-1}|\,\|_\infty}{\|A^{-1}\|_\infty}. \qquad (7.22)$$

In general, the inequality- is strict. but there is equality when $|A^{-1}| = D_1A^{-1}D_2$ for D_i of the form diag(±1), [407, 1992, Thm. 1.10], [439, 1982]. Another componentwise condition number is evaluated in Problem 7.10. We saw in Theorem 6.5 that the reciprocal of the normwise condition number for matrix inversion is the normwise relative distance to singularity. Is the same true for an appropriate componentwise condition number? The componentwise distance to singularity,

$$d_E(A) := \min\{\,\epsilon : A + \Delta A \text{ singular}, \ |\Delta A| \le \epsilon E\,\},$$

has been characterized by Rohn [877, 1989], [878, 1990] as

$$d_E(A) = \frac{1}{\max_{S_1,S_2} \rho_0(S_1A^{-1}S_2E)},$$

where the maximum is taken over all signature matrices $S_i = \mathrm{diag}(\pm 1)$ and where

$$\rho_0(X) = \max\{\,|\lambda| : \lambda \text{ is a real eigenvalue of } A\,\}.$$

This formula involves 4^n eigenproblems and thus is computationally intractable (in fact it has been shown to be NP-hard by Poljak and Rohn [836, 1993]).

Demmel [285, 1992] shows by complexity arguments that there can be no simple relationship between $d_E(A)$ and the quantity $\|\,|A^{-1}|E\,\|_\infty$, which is an upper bound for $\mu_E(A)$. He also presents evidence for the conjecture that

$$\frac{1}{\rho(|A^{-1}||A|)} \le d_{|A|}(A) \le \frac{\gamma_n}{\rho(|A^{-1}||A|)}$$

for a constant γ_n. The lower bound always holds and Demmel identifies several classes of matrices for which the upper bound holds. This conjecture is both plausible and aesthetically pleasing because $d_{|A|}(A)$ is invariant under two-sided diagonal scalings of A and $\rho(|A^{-1}||A|)$ is the minimum ∞-norm condition number achievable by such scalings, as shown by Theorem 7.8.

7.5. Extensions

The componentwise analyses can be extended in three main ways.

(1) We can use more general measures of size for the data and the solution. Higham and Higham [528, 1992] measure DA, Db, and Dx by

$$\nu_p([(\Delta a_{ij}/e_{ij})\ (\Delta b_i/f_i)]), \qquad \nu_p((\Delta x_i/g_i)),$$

where $\nu_p(A) = \left(\sum_{i,j} |a_{ij}|^p\right)^{1/p}$, $1 \le p \le \infty$, and the e_{ij}, f_i, and g_i are tolerances. They show that the corresponding backward error is given by the explicit formula

$$\left\| \left(\frac{r_j}{\|D_j \left[\begin{smallmatrix} y \\ -1 \end{smallmatrix}\right]\|_q} \right) \right\|_p,$$

where $r = b - Ay$, $D_j = \operatorname{diag}(e_{j1}, \ldots, e_{jn}, f_j)$, and $p^{-1} + q^{-1} = 1$; bounds for the corresponding condition number are also obtained. Theorem 7.3, and Theorem 7.4 with the ∞-norm, correspond to $p = \infty$ and $g_i \equiv \|x\|_\infty$. If we take $p = \infty$ and $g = |x|$, we are measuring the change in the solution in a componentwise relative sense, as $\nu_p((\Delta x_i/g_i)) = \max_i |\Delta x_i|/|x_i|$, and the condition number is [528, 1992]

$$\|\operatorname{diag}(|x_i|)^{-1}|A^{-1}|(E|x| + f)\|_\infty.$$

This latter case has also been considered by Rohn [876, 1989] and Gohberg and Koltracht [455, 1993]. It is also possible to obtain individual bounds for $\Delta x_i|/|x_i|$, $i = 1\colon n$, by refraining from taking norms in the analysis: see Chandrasekaran and Ipsen [197, 1995] and Problem 7.1.

(2) The backward error results and the perturbation theory can be extended to systems with multiple right-hand sides. For the general ν_p measure described in (1), the backward error can be computed by finding the minimum p-norm solutions to n underdetermined linear systems. For details, see Higham and Higham [528, 1992].

(3) Structure in A and b can be preserved in the analysis. For example, if A is symmetric or Toeplitz then its perturbation can be forced to be symmetric or Toeplitz too, while still using componentwise measures. References include Higham and Higham [527, 1992] and Gohberg and Koltracht [455, 1993] for linear structure, and Bartels and D. J. Higham [76, 1992] for Vandermonde structure. A symmetry-preserving normwise backward error is explored by Bunch, Demmel, and Van Loan [163, 1989], while Smoktunowicz [930, 1995] considers the componentwise case (see Problem 7.11). Symmetry-preserving normwise condition numbers are treated by D. J. Higham [526, 1995].

7.6. Numerical Stability

The backward errors examined in this chapter lead to definitions of numerical stability of algorithms for solving linear systems. Precise and formal definitions of stability can be given, but there are so many possibilities, across different problems, that to define and name each one tends to cloud the issues of interest. We therefore adopt an informal approach.

A numerical method for solving a square, nonsingular linear system $Ax = b$ is normwise *backward stable* if it produces a computed solution such that $\eta_{A,b}(\widehat{x})$ is of order the unit roundoff. How large we allow $\eta_{A,b}(\widehat{x})/u$ to be. while still declaring the method backward stable, depends on the context,. It is usually implicit in this definition that $\eta_{A,b}(\widehat{x}) = O(u)$ for all A and b, and a method that yields $\eta_{A,b}(\widehat{x}) = O(u)$ or a particular A and b is said to have performed in a normwise backward stable manner.

The significance of normwise backward stability is that the computed solution solves a slightly perturbed problem, and if the data A and b contain uncertainties bounded only normwise ($A \rightarrow A + \Delta A$ with $\|\Delta A\| = O(u\|A\|)$ and similarly for b), then $\widehat{x}$ may be the exact solution to the problem we wanted to solve, for all we know.

Componentwise backward stability is defined in a similar way: we now require the componentwise backward error $\omega_{|A|,|b|}(\widehat{x})$ to be of order u. This is a more stringent requirement than normwise backward stability. The rounding errors incurred by a met hod that is componentwise backward stable are in size and effect equivalent to the errors incurred in simply converting the data A and b to floating point numbers before the solution process begins.

If a method is normwise backward stable then, by Theorem 7.2, the forward error $\|x-\widehat{x}\|/\|x\|$ is bounded by a multiple of $\kappa(A)u$. However, a met hod can produce a solution whose forward error is bounded in this way without the normwise backward error $\eta_{A,b}(\widehat{x})$ being of order u. Hence it is useful to define a method for which $\|x - \widehat{x}\|/\|x\| = O(\kappa(A)u)$ as *normwise forward stable.* By similar reasoning involving $\omega_{|A|,|b|}(\widehat{x})$, we say a method is *componentwise forward stable* if $\|x - \widehat{x}\|/\|x\| = O(\text{cond}(A,x)u)$. Table 7.1 summarizes the definitions and the relations between them. There are several examples in this book of linear-equation-solving algorithms that are forward stable but not backward stable: Cramer's rule for $n = 2$ (§1.10.1). Gauss-Jordan elimination (§13.4), and the seminormal equations method for underdetermined systems (§20.3).

Other definitions of numerical stability can be useful (for example, *rowwise backward stability* means that $\eta_{|A|ee^T,|b|}(\widehat{x}) = O(u)$), and they will be introduced when needed.

7.7. Practical Error Bounds

Suppose we have a computed approximation $\widehat{x}$ to the solution of a linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$. What error bounds should we compute?

Table 7.1. *Backward and forward stability.*

Componentwise backward stability	$\Rightarrow$	Componentwise forward stability
$\omega_{\|A\|,\|b\|}(\widehat{x}) = O(u)$		$\dfrac{\|x-\widehat{x}\|}{\|x\|} = O(\mathrm{cond}(A,x)u)$
$\Downarrow$		$\Downarrow$
Normwise backward stability	$\Rightarrow$	Normwise forward stability
$\eta_{A,b}(\widehat{x}) = O(u)$		$\dfrac{\|x-\widehat{x}\|}{\|x\|} = O(\kappa(A)u)$

The backward error can be computed exactly, from the formulae

$$\begin{aligned}\eta_{E,f}(\widehat{x}) &= \frac{\|r\|}{\|E\|\,\|\widehat{x}\| + \|f\|},\\ \omega_{E,f}(\widehat{x}) &= \max_i \frac{|r_i|}{(E|\widehat{x}| + f)_i},\end{aligned} \tag{7.23}$$

at the cost of one or two matrix-vector products, for $r = b - A\widehat{x}$ and $E|\widehat{x}|$ The only question is what to do if the denominator is so small as to cause overflow or division by zero in the expression for $\omega_{E,f}(\widehat{x})$. This could happen, for example, when $E = |A|$ and $f = |b|$ and, for some i, $a_{ij}x_j = 0$ for all j, as is most likely in a sparse problem. LAPACK's xyyRFS ("refine solution") routines apply iterative refinement in fixed precision, in an attempt to satisfy $\omega_{|A|,|b|} \le u$. If the ith component of the denominator in (7.23) is less than safe_min/u, where safe_min is the smallest number such that 1/safe_min does not overflow, then they add $(n + 1)$ safe_min to the ith components of the numerator and denominator. A more sophisticated strategy is advocated for sparse problems by Arioli, Demmel, and Duff [24, 1989]. They suggest modifying the formula (7.23) by replacing $|b_i|$ in the denominator by $\|A(i,:)\|_1\|\widehat{x}\|_\infty$ when the ith denominator is very small. See [24, 1989] for details and justification of this strategy.

Turning to the forward error, one approach is to evaluate the forward error bound from Theorem 7.2 or Theorem 7.4, with ϵ equal to the corresponding backward error. Because x in (7.9) is unknown, we should use the modified bound

$$\frac{\|x-\widehat{x}\|_\infty}{\|\widehat{x}\|_\infty} \le \omega_{E,f}(\widehat{x}) \frac{\|\,|A^{-1}|(E|\widehat{x}| + f)\,\|_\infty}{\|\widehat{x}\|_\infty}. \tag{7.24}$$

If we have a particular E and f in mind for backward error reasons, then it is natural to use them in (7.24). However, the size of the forward error bound varies with E and f, so it is natural to ask which choice minimizes the bound.

Lemma 7.9. *The upper bound in (7.23) is at least as large as the upper bound in*

$$\frac{\|x-\widehat{x}\|_\infty}{\|\widehat{x}\|_\infty} \le \frac{\|\,|A^{-1}||r|\,\|_\infty}{\|\widehat{x}\|_\infty}, \qquad (7.25)$$

and is equal to it when $E|\widehat{x}|+f$ *is a multiple of* $|r|$.

Proof. First note that $r = b - A\widehat{x}$ implies $|x-\widehat{x}| \le |A^{-1}||r|$, which implies (7.25). Now, for $z \ge 0$,

$$|A^{-1}||r| = |A^{-1}|\left(z_i\frac{|r_i|}{z_i}\right) \le \max_i \frac{|r_i|}{|z_i|}|A^{-1}|z,$$

with equality if z is a multiple of r. Taking $z = E|\widehat{x}|+f$ gives

$$|A^{-1}||r| \le \omega_{E,f}(\widehat{x})|A^{-1}|(E|\widehat{x}|+f),$$

with equality when $E|\widehat{x}|+f$ is a multiple of $|r|$. The truth of this statement is preserved when ∞-norms are taken, so the result follows. □

Since the bound (7.25) is obtained by taking absolute values in the equation $x-\widehat{x} = A^{-1}r$, it is clearly the smallest possible such bound subject to ignoring signs in A^{-1} and r. It is reasonable to ask why we do not take $\|A^{-1}r\|_\infty/\|$ as our error bound. (Theoretically it is an exact bound!) The reason is that we cannot compute r or $\|A^{-1}r\|_\infty$ exactly. In place of r we compute $\widehat{r} = fl(b-A\widehat{x})$, and

$$\widehat{r} = r + \Delta r, \qquad |\Delta r| \le \gamma_{n+1}(|A||\widehat{x}|+|b|). \qquad (7.26)$$

Therefore a strict bound. and one that should be used in practice in place of (7.25), is

$$\frac{\|x-\widehat{x}\|_\infty}{\|\widehat{x}\|_\infty} \le \frac{\|\,|A^{-1}|(|\widehat{r}|+\gamma_{n+1}(|A||\widehat{x}|+|b|)\,\|_\infty}{\|\widehat{x}\|_\infty}. \qquad (7.27)$$

This forward error bound is estimated and returned by LAPACK's xyyRFS routines. For details on how this is done without computing A^{-1}, see Chapter 14.

The LAPACK linear equation solvers estimate only one condition number: the standard condition number $\kappa_1(A)$ (or, rather, its reciprocal, referred to as rcond), which is returned by the xyyCON routines.

7.8. Perturbation Theory by Calculus

The perturbation results in this book are all derived algebraically, without any use of derivatives. Calculus can also be used to derive perturbation bounds, often in a straight forward fashion.

As a simple example, consider a linear system $A(t)z(t) = b(t)$, where $A(t) \in \mathbb{R}^{n\times n}$ and $x(t), b(t) \in \mathbb{R}^n$ are assumed to be continuously differentiable functions of t. Differentiating gives

$$\dot{A}(t)x(t) + A(t)\dot{x}(t) = \dot{b}(t),$$

or, dropping the t arguments,

$$\dot{x} = -A^{-1}\dot{A}x + A^{-1}\dot{b}.$$

Taking norms, we obtain

$$\frac{\|\dot{x}\|}{\|x\|} \le \|A^{-1}\|\,\|\dot{A}\| + \|A^{-1}\|\frac{\|\dot{b}\|}{\|x\|} \le \kappa(A)\left(\frac{\|\dot{A}\|}{\|A\|} + \frac{\|\dot{b}\|}{\|b\|}\right).$$

This bound shows that $\kappa(A)$ is a key quantity in measuring the sensitivity of a linear system. A componentwise bound could have been obtained just as easily.

We normally express perturbations of the data in the form $A \to A + \Delta A$. To use the calculus framework we can take $A(0)$ as the original matrix A and write $A + \Delta A := A(\epsilon) = A(0) + \epsilon\dot{A}(0) + O(\epsilon^2)$, but the perturbation bound then becomes a first-order one.

The calculus technique is a useful addition to the armoury of the error analyst (it is used by Golub and Van Loan [470, 1989], for example), but the algebraic approach is preferable for deriving rigorous perturbation bounds of the standard forms.

7.9. Notes and References

This chapter draws on the survey paper Higham [558, 1994].

Theorem 7.3 is due to Oettli and Prager [802, 1964], and predates the normwise backward error result Theorem 7.1 of Rigal and Gaches [873, 1967]. In addition to Theorem 7.1, Rigal and Gaches give a more general result based on norms of blocks that includes Theorems 7.3 and 7.1 as special cases. Theorem 7.1 is also obtained by Kovarik [672, 1976].

Theorems 7.1 and 7.3 both remain valid when A is rectangular. Componentwise backward error for rectangular A was considered by Oettli, Prager, and Wilkinson [803, 1965], but their results are subsumed by those of Oettli and Prager [802, 1964] and Rigal and Gaches [873, 1967].

For a linear system $Ax = b$ subject to componentwise perturbations, Oettli [801, 1965] shows how linear programming can be used to obtain bounds on the components of x when all solutions of the perturbed system lie in the same orthant. Cope and Rust [244, 1979] extend this approach by showing, in general, how to bound all the solutions that lie in a given orthant. This type of analysis can also be found in the book by Kuperman [681, 1971], which includes an independent derivation of Theorem 7.3. See also Hartfiel [505, 1980].

Theorem 7.4 is a straightforward generalization of a result of Skeel [919, 1979, Thms. 2.1 and 2.2]. It is clear from Bauer's comments in [80, 1966] that the bound (7.9), with $E = |A|$ and $f = |b|$, was known to him, though he does not state the bound. This is the earliest reference we know in which componentwise analysis is used to derive forward perturbation bounds.

Theorem 7.8 is from Bauer [79, 1963]. Bauer actually states that equality holds in (7.21) for any A, but his proof of equality is valid only when $|A^{-1}||A|$ and $|A||A^{-1}|$ have positive Perron vectors. Businger [168, 1968] proves that a sufficient condition for the irreducibility condition of Theorem 7.8 to hold (which, of course, implies the positivity of the Perron vectors) is that there do not exist permutations P and Q such that PAQ is in block triangular form.

Theorem 7.7 is from Stewart and Sun [954, 1990, Thm. 4.3.5].

Further results on scaling to minimize the condition number $\kappa(A)$ are given by Forsythe and Straus [386, 1955], Bauer [81, 1969], Golub and Varah [465, 1974], McCarthy and Strang [742, 1974], Shapiro [913, 1982]. [914, 1985], [915, 1991], and Watson [1067, 1991].

Chan and Foulser [193, 1988] introduce the idea of "effective conditioning" for linear systems, which takes into account the projections of b onto the range space of A. See Problem 7.5, and for an application to partial differential equations see Christiansen and Hansen [208, 1994].

For an example of how definitions of numerical stability for linear equation solvers can be extended to incorporate structure in the problem, see Bunch [162, 1987].

An interesting application of linear system perturbation analysis is to Markov chains. A discrete-time Markov chain can be represented by a square matrix P, where p_{ij} is the probability of a transition from state i to state j. Since state i must lead to some other state. $\sum_j p_{ij} = 1$, and these conditions can be writ ten in matrix vector form as

$$Pe = e. \tag{7.28}$$

A nonnegative matrix satisfying (7.28) is called a *stochastic* matrix. The initial state of the Markov chain can be defined by a vector z^T, where z_i, denotes the probability that the ith state of the chain is occupied. Then the state of the chain at the next time unit is given by z^TP. The *steady state* or *stationary vector* of the chain is given by

$$\pi^T = \lim_{k\to\infty} z^T P^k.$$

An important question is the sensitivity of the individual components of the steady-state vector to perturbations in P. This is investigated, for example.

by Ipsen and Meyer [605, 1994], who measure the perturbation matrix normwise, and by O'Cinneide [800, 1993], who measures the perturbation matrix componentwise. For a matrix-oriented development of Markov chain theory see Berman and Plemmons [94, 1994].

It is possible to develop probabilistic perturbation theory for linear systems and other problems by making assumptions about the statistical distribution of the perturbations. We do not consider this approach here (though see Problem 7.13), but refer the interested reader to the papers by Fletcher [376, 1985], Stewart [948, 1990], and Weiss, Wasilkowski, Wozniakowski, and Shub [1073, 1986].

Problems

7.1. Under the conditions of Theorem 7.4, show that

$$|x-y| \le \epsilon(I-\epsilon|A^{-1}|E)^{-1}|A^{-1}|(f+E|x|).$$

Hence derive a first-order bound for $|x_i - y_i|/|x_i|$.

7.2. Let $Ax = b$, where $A \in \mathbb{R}^{n\times n}$. Show that for any vector y and any subordinate matrix norm,

$$\frac{\|r\|}{\|A\|\,\|x\|} \le \frac{\|x-y\|}{\|x\|} \le \kappa(A)\frac{\|r\|}{\|A\|\,\|x\|},$$

where the residual $r = b - Ay$. Interpret this result.

7.3. Prove (7.13) and deduce (7.12).

7.4. Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite and let $A = DHD$, where $D = \mathrm{diag}(a_{ii}^{1/2})$ (this is the scaling used in Corollary 7.6). Show that $\mathrm{cond}(H) \le \kappa_\infty(H) \le n\,\mathrm{cond}(H)$.

7.5. (Chan and Foulser [193, 1988]) Let $A \in \mathbb{R}^{n\times n}$ have the SVD $A = U\Sigma V^T$, where $\Sigma = \mathrm{diag}(\sigma_i)$, $\sigma_1 \ge \cdots \ge \sigma_n$, and define the projection matrix $P_k := U_k U_k^T$, where $U_k = U(:, n+1-k\colon n)$. Show that if $Ax = b$ and $A(x + \Delta x) = (b + \Delta b)$ then

$$\frac{\|\Delta x\|_2}{\|x\|_2} \le \frac{\sigma_{n+1-k}}{\sigma_n}\frac{\|b\|_2}{\|P_k b\|_2}\frac{\|\Delta b\|_2}{\|b\|_2}, \qquad k = 1\colon n.$$

What is the interpretation of this result?

7.6. (a) For the choice of tolerances $E = |A|ee^T$, $f = |b|$, corresponding to a row-wise backward error, show that

$$\mathrm{cond}(A)\frac{\|x\|_1}{\|x\|_\infty} \le \mathrm{cond}_{E,f}(A,x) \le 2\,\mathrm{cond}(A)\frac{\|x\|_1}{\|x\|_\infty}.$$

(b) For $E = ee^T|A|$ and $f = \|b\|_\infty e$, corresponding to a columnwise backward error, show that

$$\|A^{-1}\|_\infty \frac{\|\,|A||x|\,\|_1}{\|x\|_\infty} \le \operatorname{cond}_{E,f}(A,x) \le 2\|A^{-1}\|_\infty \frac{\|\,|A||x|\,\|_1}{\|x\|_\infty}.$$

7.7. Show that

$$\omega_{|A|,|b|}(y) \le \omega_{|A|,0}(y) \le \frac{2\omega_{|A|,|b|}(y)}{1-\omega_{|A|,|b|}(y)},$$

$$\eta_{A,b}(y) \le \eta_{|A|,0}(y) \le \frac{2\eta_{A,b}(y)}{1-\eta_{A,b}(y)}.$$

7.8. Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. A componentwise condition number for the problem of computing c^Tx, where $Ax = b$, can be defined by

$$\chi_{E,f}(A,x) := \lim_{\epsilon\to 0} \sup\biggl\{ \frac{|c^T\Delta x|}{\epsilon|c^Tx|} : (A+\Delta A)(x+\Delta x) = b+\Delta b,$$
$$|\Delta A| \le \epsilon E,\ \ |\Delta b| \le \epsilon f \biggr\}.$$

Obtain an explicit formula for $\chi_{E,f}(A, x)$. Show that $\chi_{E,f}(A, x) \ge 1$ if $E = |A|$ or $f = |b|$. Derive the corresponding normwise condition number $\psi_{E,f}(A,x)$, in which the constraints are $\|\Delta A\|_2 \le \epsilon\|E\|_2$ and $\|\Delta b\|_2 \le \epsilon\|f\|_2$.

7.9. (Bauer [79, 1963]) Let $A, B, C \in \mathbb{R}^{n\times n}$. (a) Prove that if B and C have positive elements then

$$\min_{D_1,D_2\in\mathcal{D}_n} \|D_1BD_2\|_\infty \|D_2^{-1}CD_1^{-1}\|_\infty = \rho(BC),$$

where $\mathcal{D}_n = \{\operatorname{diag}(d_i) : d_i > 0,\ i = 1{:}n\}$. (Hint: consider $D_1 = \operatorname{diag}(x_1)^{-1}$ and $D_2 = \operatorname{diag}(Cx_1)$, where $x_1 > 0$ is a right Perron vector of BC: $BCx_1 = \rho(BC)x_1$.)

(b) Deduce that if $|A|$ and $|A^{-1}|$ have positive entries, then

$$\min_{D_1,D_2\in\mathcal{D}_n} \kappa_\infty(D_1AD_2) = \rho(|A||A^{-1}|).$$

(c) Show that for any nonsingular A,

$$\inf_{D_1,D_2\in\mathcal{D}_n} \kappa_\infty(D_1AD_2) \le \rho(|A||A^{-1}|).$$

(d) Strengthen (b) by showing that for any nonsingular A such that $|A||A^{-1}|$ and $|A^{-1}||A|$ are irreducible,

$$\min_{D_1,D_2\in\mathcal{D}_n} \kappa_\infty(D_1AD_2) = \rho(|A||A^{-1}|).$$

(e) What can you deduce about $\min_{D_1,D_2\in\mathcal{D}_n}\kappa(D_1AD_2)$ for the 1- and 2-norms?

7.10. (Bauer [80, 1966, p. 413], Rohn [876, 1989]) We can modify the definition of $mE(A)$ in (7.22) by measuring DX componentwise relative to X, giving

$$\mu'_E(A) := \lim_{\epsilon\to 0}\sup\Big\{\max_{i,j}\frac{|\Delta x_{ij}|}{|x_{ij}|} : |\Delta A|\le \epsilon E\Big\}$$

(where $X = A^{-1}$ and $X+\Delta X = (A+\Delta A)^{-1}$). Show that

$$\mu'_E(A) = \max_{ij}\frac{\big(|A^{-1}||E||A^{-1}|\big)_{ij}}{\big(|A^{-1}|\big)_{ij}}.$$

7.11. Let $A\in\mathbb{R}^{n\times n}$ lbe symmetric and let y be an approximate solution to $Ax = b$. If y has a small backward error, so that y solves a nearby system. does it follow that y solves a nearby *symmetric* system? This problem answers the question for both the normwise and componentwise relative backward errors.

(a) (Bunch, Demmel, and Van Loan [163, 1989]) Show that if $(A+G)y = b$ then there exists $H = H^T$ such that $(A + H)y = b$ with $\|H\|_2 \le \|G\|_2$ and $\|H\|_F \le\sqrt{2}\|G\|_F$. (This result does not require $A = A^T$.)

(b) (Smoktunowicz [930, 1995]) Show that if A is symmetric and diagonally dominant and $(A + G)y = b$ with $|G| \le \epsilon|A|$, then there exists $H = H^T$ such that $(A + H)y = b$ with $|H| \le 3\epsilon|A|$. (For a general symmetric A there may not exist such an H, as is easily shown by a 2 × 2 example [527, 1992].)

(c) (Smoktunowicz [930, 1995]) Show that if A is symmetric positive definite and $(A + G)y = b$ with $|G| \le \epsilon|A|$ then there exists $H = H^T$ such that $(A + H)y = b$ with $|H| \le (2n-1)\epsilon|A|$.

7.12. Suppose that $A\in\mathbb{R}^{n\times n}$ has w_i nonzeros in its ith row, $i = 1{:}n$. Show that the inequality (7.27) can be replaced by

$$\frac{\|x-\widehat{x}\|_\infty}{\|\widehat{x}\|_\infty}\le\frac{\|\,|A^{-1}|(|\widehat{r}|+\Gamma(|A||\widehat{x}|+|b|)\,\|_\infty}{\|\widehat{x}\|_\infty},$$

where $\Gamma = \mathrm{diag}(\gamma_{w_i+1})$. This bound is potentially much smaller than (7.27) for large, sparse matrices.

7.13. (D. J. Higham, after Fletcher [376, 1985]) Suppose the nonsingular, square matrix A is perturbed to $A + DA$ and b to $b + Db$. Then, to first order, the solution of $Ax = b$ is perturbed to $x + Dx$, where

$$\Delta x = -A^{-1}\Delta Ax + A^{-1}\Delta b.$$

Suppose that the perturbations have the form

$$\Delta a_{ij} = \epsilon_{ij} e_{ij}, \quad \Delta b_i = \delta_i f_i,$$

where the ϵ_{ij} and δi are independent random variables, each having zero mean and variance s^2. (As usual, the matrix E and vector f represent fixed tolerances.) Let ε denote the expected value.

(a) Show that

$$\mathcal{E}(\|\Delta x\|_2^2) = \sigma^2 \| \, [A^{-1}][E][x] + [A^{-1}][f] \, \|_1,$$

where square brackets denote the operation of elementwise squaring: $[B]_{ij} = b_{ij}^2$.

(b) Hence explain why

$$\text{condexp}(A, x) := \frac{\left(\| [A^{-1}][E][x] + [A^{-1}][f] \|_1 \right)^{1/2}}{\|x\|_2}$$

may be regarded as an "expected condition number" for the linear system $Ax = b$.

(c) For the case where $e_{ij} \equiv \|A\|_2$ and $f_j \equiv \|b\|_2$. compare condexp(A, x) with the "worst-case" condition number $k_{A,b}(A, x)$ for the 2-norm.

7.14. (Horn and Johnson [581, 1991, p. 331]) Prove that for any nonsingular $A \in \mathbb{R}^{n \times n}$,

$$\inf_{D_1, D_2 \in \mathcal{D}_n} \kappa_2(D_1 A D_2) \geq \|A \circ A^{-T}\|_2,$$

where $\circ$ is the Hadamard product $(A \circ B = (a_{ij} b_{ij}))$ and $\mathcal{D}_n$ is defined as in Problem 7.9. (Hint: use the inequality $\|A \circ B\|_2 \leq \|A\|_2 \|B\|_2$.) Discuss the attainability of this bound.

Chapter 8

Triangular Systems

> *In the end there is left the coefficient of one unknown and the constant term.*
> *An elimination between this equation and*
> *one from the previous set that contains two unknowns*
> *yields an equation with the coefficient of*
> *another unknown and another constant term, etc.*
> *The quotient of the constant term by the unknown*
> *yields the value of the unknown in each case.*
>
> -JOHN V. ATANASOFF, *Computing Machine for the Solution of*
> *Large Systems of Linear Algebraic Equations (1940)*

> The solutions of triangular systems are usually computed to high accuracy.
> *This fact . . . cannot be proved in genera/, for counter examples exist.*
> *However, it is true of many special kinds of triangular matrices and*
> *the phenomenon has been observed in many others.*
> *The practical consequences of this fact cannot be over-emphasized.*
>
> -G. W. STEWART, *Introduction to Matrix Computations* (1973)

> *In practice one almost invariably finds that*
> *if* L *is ill-conditioned,* so that $\|L\|\,\|L^{-1}\| \gg 1$,
> *then the computed solution of Lx = b (or the computed inverse)*
> *is far more accurate than [standard norm bounds] would suggest.*
>
> -J. H. WILKINSON, *Rounding Errors in Algebraic Processes* (1963)

Triangular systems play a fundamental role in matrix computations. Many methods are built on the idea of reducing a problem to the solution of one or more triangular systems, including virtually all direct methods for solving linear systems. On serial computers triangular systems are universally solved by the standard back and forward substitution algorithms. For parallel computation there are several alternative methods, one of which we analyse in §8.4.

Backward error analysis for the substitution algorithms is straightforward and the conclusion is well known: the algorithms are extremely stable. The behaviour of the forward error, however, is intriguing, because the forward error is often surprisingly small --much smaller than we would predict from the normwise condition number κ, or, sometimes, even the componentwise condition number cond. The quotes from Stewart and Wilkinson at the start of this chapter emphasize the high accuracy that is frequently observed in practice. The analysis we give in this chapter provides a partial explanation for the observed accuracy of the substitution algorithms. In particular, it reveals three important but nonobvious properties:

- the accuracy of the computed solution from substitution depends strongly on the right-hand side:
- a triangular matrix may be much more or less ill conditioned than its transpose; and
- the use of pivoting in LU, QR, and Cholesky factorizations can greatly improve the conditioning of a resulting triangular system.

As well as deriving backward and forward error bounds, we show how to compute upper and lower bounds for the inverse of a triangular matrix.

8.1. Backward Error Analysis

Recall that for an upper triangular matrix $U \in \mathbb{R}^{n\times n}$ the system $Ux = b$ can be solved using the formula $x_i = (b_i - \sum_{j=i+1}^{n} u_{ij}x_j)/u_{ii}$, which yields the components of x in order from last to first.

Algorithm 8.1 (back substitution). Given a nonsingular upper triangular matrix $U \in \mathbb{R}^{n\times n}$ this algorithm solves the system $Ux = b$.

```
$x_n = b_n/u_{nn}$
for $i = n - 1:-1:1$
    $s = b_i$
    for $j = i + 1:n$
        $s = s - u_{ij}x_{ij}$
        end
        $x_i = s/u_{ii}$
    end
```

We will not state the analogous algorithm for solving a lower triangular system, forward substitution. All the results below for back substitution have obvious analogues for forward substitution. Throughout this chapter T denotes a matrix that can be upper or lower triangular.

To analyse the errors in substitution we need the following lemma.

Lemma 8.2. *Let* $y = (c - \sum_{i=1}^{k-1} a_i b_i)/b_k$ *be evaluated in floating point arithmetic according to*

```
    $s = c$
    for $i$ = 1:1$k$-1
        $s = s - a_i b_i$
    end
    $y = s/bk$
```

Then the computed $\widehat{y}$ *satisfies*

$$b_k \widehat{y}(1+\theta_k) = c - \sum_{i=1}^{k-1} a_i b_i (1+\theta_i), \tag{8.1}$$

where $|q_i| \le g_i = iu/(1 - iu)$.

Proof. Analysis very similar to that leading to (3.2) shows that $\widehat{s} := fl(c - \sum_{i=1}^{k-1} a_i b_i)$ satisfies

$$\widehat{s} = c(1+\delta_1)\dots(1+\delta_{k-1}) - \sum_{i=1}^{k-1} a_i b_i (1+\epsilon_i)(1+\delta_i)\dots(1+\delta_{k-1}),$$

where $|\epsilon_i|, |\delta_i| \le u$. The final division yields, using (2.5), $\widehat{y} = fl(\widehat{s}/b_k) = \widehat{s}/(b_k(1+\delta_k))$, $|\delta_k| \le u$, so that, after dividing through by $(1+\delta_1)\dots(1+\delta_{k-1})$, we have

$$b_k \widehat{y} \frac{1+\delta_k}{(1+\delta_1)\dots(1+\delta_{k-1})} = c - \sum_{i=1}^{k-1} a_i b_i \frac{1+\epsilon_i}{(1+\delta_1)\dots(1+\delta_{i-1})}.$$

The result is obtained on invoking Lemma 3.1. ☐

Two remarks are in order. First, we chose the particular form of (8.1), in which c is not perturbed, in order to obtain a backward error result for $Ux = b$ in which b is not perturbed. Second, we carefully kept track of the terms $1 + \delta_i$ in the proof, so as to obtain the best possible constants. Direct application of the lemma to Algorithm 8.1 yields a backward error result.

Theorem 8.3. *The computed solution* $\widehat{x}$ *from Algorithm* 8.1 *satisfies*

$$(U+\Delta U)\widehat{x}=b, \qquad |\Delta u_{ij}| \le \begin{cases} \gamma_{n-i+1}|u_{ii}|, & i=j, \\ \gamma_{|i-j|}|u_{ij}|, & i \ne j. \end{cases} \qquad \square$$

Theorem 8.3 holds only for the particular ordering of arithmetic operations used in Algorithm 8.1. A result that holds for any ordering is a consequence of the next lemma.

Lemma 8.4. *If* $y = (c - \sum_{i=1}^{k-1} a_i b_i)/b_k$ *is evaluated in floating point arithmetic, then. no matter what the order of evaluation,*

$$b_k\widehat{y}(1+\theta_k^{(0)}) = c - \sum_{i=1}^{k-1} a_i b_i(1+\theta_k^{(i)}),$$

where $|\theta_k^{(i)}| \le \gamma_k$ *for all i. If* $b_k = 1$, *so that there is no division, then* $|\theta_k^{(i)}| \le \gamma_{k-1}$ *for all i.*

Proof. The result is not hard to see after a little thought , but a formal proof is tedious to write down. Note that the ordering used in Lemma 8.2 is the one for which this lemma is least obvious! The last part of the lemma is useful when analysing unit lower triangular systems, and in various other contexts. $\square$

Theorem 8.5. *Let the triangular system TX = b, where* $T \in \mathbb{R}^{n\times n}$ *is nonsingular, be solved by substitution, with any ordering. Then the computed solution* $\widehat{x}$ *satisfies*

$$(T+\Delta T)\widehat{x}=b, \qquad |\Delta T| \le \gamma_n|T|. \qquad \square$$

In technical terms, this result says that $\widehat{x}$ has a tiny componentwise relative backward error. In other words, the backward error is about as small as we could possibly hope.

In most of the remaining error analyses in this book, we will derive results that, like the one in Theorem 8.5, do not depend on the ordering of the arithmetic operations. Results of this type are more general, usually no less informative. and easier to derive, than ones that depend on the ordering. However, it is important to realise that *the actual error does depend on the ordering,* possibly strongly so for certain data. This point is clear from Chapter 4 on summation.

8.2. Forward Error Analysis

From Theorems 8.5 and 7.4 there follows the forward error bound

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \le \frac{\operatorname{cond}(T,x)\gamma_n}{1-\operatorname{cond}(T)\gamma_n},$$

where

$$\operatorname{cond}(T,x) = \frac{\|\,|T^{-1}||T||x|\,\|_\infty}{\|x\|_\infty}, \quad \operatorname{cond}(T) = \|\,|T^{-1}||T|\,\|_\infty.$$

This bound can, of course, be arbitrarily smaller than the corresponding bound involving $\kappa_\infty(T) = \|T\|_\infty \|T^{-1}\|_\infty$ for the reasons explained in Chapter 7. For further insight, note that, in terms of the traditional condition number, $\kappa(T)$, ill conditioning of a triangular matrix stems from two possible sources: variation in the size of the diagonal elements and rows with off-diagonal elements which are large relative to the diagonal elements. Significantly, because of its row scaling invariance, cond(T, x) is susceptible only to the second source.

Despite its pleasing properties, cond(T, x) can be arbitrarily large. This is illustrated by the upper triangular matrix

$$U(\alpha) = (u_{ij}), \qquad u_{ij} = \begin{cases} 1, & i=j, \\ -\alpha, & i<j, \end{cases} \tag{8.2}$$

for which

$$\left(U(\alpha)^{-1}\right)_{ij} = \begin{cases} 1, & i=j, \\ \alpha(1+\alpha)^{j-i-1}, & j>i. \end{cases} \tag{8.3}$$

We have cond($U(a)$,e) = cond($U(a)$) $\sim 2a^{n-1}$ as $a \to \infty$. Therefore WC cannot assert that *all* triangular systems are solved to high accuracy. Nevertheless, for any T there is always at least one system for which high accuracy is obtained: the system $TX = e_1$ if T is upper triangular, or $TX = e_n$ if T is lower triangular. In both cases cond(T, x) = 1, and the solution comprises the computation of just a single scalar reciprocal.

To gain further insight we consider special classes of triangular matrices, beginning with one produced by certain standard factorizations with pivoting. In all the results below, the triangular matrices are assumed to be $n \times n$ and nonsingular, and $\widehat{x}$ is the computed solution from substitution.

Lemma 8.6. *Suppose the upper triangular matrix* $U \in \mathbb{R}^{n\times n}$ *satisfies*

$$|u_{ii}| \ge |u_{ij}| \quad \textit{for all } j > i. \tag{8.4}$$

Then the unit upper triangular matrix $W = |U^{-1}||U|$ *satisfies* $w_{ij} \le 2^{j-i}$ *for all* $j > i$.

Proof. We can write $W = |V^{\,1}||V|$ where $V = D^{-1}\ U$ and $D = \mathrm{diag}(u_{ii})$. The matrix V is unit upper triangular with) $|v_{ij}| \le 1$, and it is easy to show that $|(V^{-1})_{ji}| \le 2^{j-i-1}$ for $j > i$. Thus. for $j > i$,

$$w_{ij} = \sum_{k=i}^{j} \big|(V^{-1})_{ik}\big||v_{kj}| \le 1 + \sum_{k=i+1}^{j} 2^{k-i-1} \cdot 1 = 2^{j-i}. \qquad \square$$

Theorem 8.7. *Under the conditions of Lemma 8.6, the computed solution* $\widehat{x}$ *to* $Ux = b$ *obtained by substitution satisfies*

$$|x_i - \widehat{x}_i| \le 2^{n-i+1}\gamma_n \max_{j \ge i} |\widehat{x}_j|, \qquad i = 1\colon n.$$

Proof. From Theorem 8.5 we have

$$|x - \widehat{x}| = |U^{-1}\Delta U\widehat{x}| \le \gamma_n |U^{-1}||U||\widehat{x}|.$$

Using Lemma 8.6 we obtain

$$|x_i - \widehat{x}_i| \le \gamma_n \sum_{j=i}^{n} w_{ij}|\widehat{x}_j| \le \gamma_n \max_{j\ge i} |\widehat{x}_j| \sum_{j=i}^{n} 2^{j-i} \le 2^{n-i+1}\gamma_n \max_{j \ge i} |\widehat{x}_j|. \quad \square$$

Lemma 8.6 shows that for matrices satisfying (8.4), cond(T) is bounded for fixed n, no matter how large $\kappa(T)$. The bounds for $|x_i - \widehat{x}_i|$ in Theorem 8.7, although large if n is large and i is small. decay exponentially with increasing thus, later components of x are always computed to high accuracy relative to the elements already computed.

Analogues of Lemma 8.6 and Theorem 8.7 hold for lower triangular L satisfying

$$|l_{ii}| \ge |l_{ij}| \text{ for all } j < i. \tag{8.5}$$

Note, however, that if the upper triangular matrix T satisfies (8.4) then T^T does not necessarily satisfy (8.5). In fact, cond(T^T) can he arbitrarily large, as shown by the example

$$T = \begin{bmatrix} 1 & 1 & 0 \\ 0 & \epsilon & \epsilon \\ 0 & 0 & 1 \end{bmatrix},$$

$$\mathrm{cond}(T) = 5, \quad \mathrm{cond}(T^T) = 1 + \frac{2}{\epsilon}.$$

An important conclusion is that a triangular system $Tx = b$ can be much more or less ill conditioned than the system $T^Ty = c$, even if T satisfies (8.4).

Theorem 8.7, or its lower triangular analogue, is applicable to

- the lower triangular matrices from Gaussian elimination with partial pivoting or complete pivoting;
- the upper triangular matrices from Gaussian elimination with complete pivoting;
- the upper triangular matrices from the Cholesky and ***QR*** factorizations with complete pivoting and column pivoting, respectively.

Next, we consider triangular T satisfying

$$t_{ii} > 0, \ t_{ij} \leq 0 \ \text{for all} \ i \neq j.$$

It is easy to see that such a matrix has an inverse with nonnegative elements, and hence is an hi-matrix (for definitions of an hi-matrix see Appendix B). Associated with any square matrix A is the *comparison matrix:*

$$M(A) = (m_{ij}), \qquad m_{ij} = \begin{cases} |a_{ii}|, & i = j, \\ -|a_{ij}|, & i \neq j. \end{cases} \tag{8.6}$$

For any nonsingular triangular T, $M(T)$ is an M-matrix. Furthermore, it is easy to show that $|T^{-1}| \leq M(T)^{-1}$ (see Theorem 8.11).

The following result shows that among all matrices R such that $|R| = |T|$, $R = M(T)$ is the one that maximizes $\mathrm{cond}(R,x)$.

Lemma 8.8. *For any triangular T,*

$$\mathrm{cond}(T,x) \leq \mathrm{cond}(M(T),x) = \|\,\big(2M(T)^{-1}\,\mathrm{diag}(|t_{ii}|) - I\big)|x|\,\|_\infty \Big/ \|x\|_\infty.$$

Proof. The inequality follows from $|T^{\ 1}| \leq M(T)^{-1}$, together with $|T| = |M(T)|$. Since $M(T)^{-1} \geq 0$, we have

$$\begin{aligned} |M(T)^{-1}||M(T)| &= M(T)^{-1} \ (2\mathrm{diag}(|t_{ii}|) - M(T)) \\ &= 2M(T)^{-1} \ \mathrm{diag}(|t_{ii}|) - I, \end{aligned}$$

which yields the equality. □

If $T{=}M(T)$ has unit diagonal then, using Lemma 8.8,

$$\mathrm{cond}(T) = \mathrm{cond}(T,e) = \|2T^{-1} - I\|_\infty \approx 2\frac{\kappa(T)}{\|T\|_\infty}.$$

This means, for example, that the system $U(1)x = b$ (see (8.2)), where $x = e$, is about as ill conditioned with respect to componentwise relative perturbations in $U(1)$ as it is with respect to normwise perturbations in $U(1)$.

The next result gives a forward error bound for substitution that is proved directly, without reference to the backward error result Theorem 8.5 (indeed. it cannot be obtained from that result!). The bound can be very weak, because $\|M(T)^{-1}\|$ can be arbitrarily larger than $\|T^{-1}\|$ (see Problem 8.2), but it yields a pleasing bound in the special case described in the corollary.

Theorem 8.9. *The computed solution* $\widehat{x}$ *obtained from substitution applied to the triangular system* $Tx = b$ *of order* n *satisfies*

$$|x-\widehat{x}| \le \bigl((n^2+n+1)u + O(u^2)\bigr)M(T)^{-1}|b|.$$

Proof. Without loss of generality, suppose $T = L$, is lower triangular. The proof is by induction on the components of x. The result clearly holds for the first component. Assume that it holds for the first $n - 1$ components. An analogue of Lemma 8.3 shows that

$$l_{nn}\widehat{x}_n = b_n(1+\theta_{n+1}^{(0)}) - \sum_{j=1}^{n-1} l_{nj}\widehat{x}_j(1+\theta_{n+1}^{(j)}),$$

where $|\theta_{n+1}^{(j)}| \le \gamma_{n+1}$ for all j. Subtracting from $l_{nn}x_n = b_n - \sum_{j=1}^{n-1} l_{nj}x_j$ gives

$$l_{nn}(x_n-\widehat{x}_n) = -b_n\theta_{n+1}^{(0)} - \sum_{j=1}^{n-1} l_{nj}(x_j-\widehat{x}_j) + \sum_{j=1}^{n-1} l_{nj}\widehat{x}_j\theta_{n+1}^{(j)},$$

so that

$$|x_n-\widehat{x}_n| \le \gamma_{n+1}\frac{|b_n|}{|l_{nn}|} + \sum_{j=1}^{n-1}\frac{|l_{nj}|}{|l_{nn}|}|x_j-\widehat{x}_j| + \gamma_{n+1}\sum_{j=1}^{n-1}\frac{|l_{nj}|}{|l_{nn}|}|\widehat{x}_j|. \tag{8.7}$$

Write

$$M(L) = \begin{bmatrix} M_{11} & 0 \\ -m^T & m_{nn} \end{bmatrix}, \qquad M(L)^{-1} = \begin{bmatrix} M_{11}^{-1} & 0 \\ m_{nn}^{-1}m^TM_{11}^{-1} & m_{nn}^{-1} \end{bmatrix}.$$

Then the inductive assumption can be written as $|\widehat{x}(1{:}n-1)-x(1{:}n-1)| \le \mu_{n-1}M_{11}^{-1}|b(1{:}n-1)|$, which implies $(1{:}n-1)| \le (\mu_{n-1}+1)M_{11}^{-1}|b(1{:}n-1)|$. Hence (8.7) gives

$$\begin{aligned} |x_n-\widehat{x}_n| &\le \gamma_{n+1}m_{nn}^{-1}|b_n| + \mu_{n-1}m_{nn}^{-1}m^TM_{11}^{-1}|b(1{:}n-1)| \\ &\quad + \gamma_{n+1}(\mu_{n-1}+1)m_{nn}^{-1}m^TM_{11}^{-1}|b(1{:}n-1)| \\ &\le \bigl(\mu_{n-1}+\gamma_{n+1}(\mu_{n-1}+1)\bigr)\bigl(M(L)^{-1}|b|\bigr)_n. \end{aligned}$$

We have the recurrence $\mu_k = (1+\gamma_{k+1})\mu_{k-1} + \gamma_{k+1} \le (1+\gamma_{n+1})\mu_{k-1} + \gamma_{n+1}$, $\mu_0 = u$, which yields

$$\mu_n \le (1+\gamma_{n+1})^n u + \big((1+\gamma_{n+1})^n - 1\big) \le (n^2+n+1)u + O(u^2). \qquad \square$$

Corollary 8.10. *The computed solution obtained from substitution applied to the triangular system $Tx = b$ of order n, where $T = M(T)$ and $b \ge 0$, satisfies*

$$|x - \widehat{x}| \le \big((n^2+n+1)u + O(u^2)\big)|x|. \qquad \square$$

Corollary 8.10 shows that, when T is an M-matrix and the right-hand side is nonnegative, the solution is obtained to high relative accuracy in every component. The reason for the high accuracy is that for such a system there are no subtractions of like-signed numbers, so that each x_i is computed as a sum of nonnegative quantities. A consequence of the corollary is that the inverse of a triangular M-matrix can be computed to high relative accuracy.

Triangular systems of the type in Corollary 8.10 occur in linear equations obtained by discretizing certain elliptic partial differential equations, such as the Poisson equation on a rectangle, with zero boundary conditions and a positive forcing function: these problems yield symmetric positive definite M-matrices, and the LU factors of an M-matrix are themselves M-matrices. Such systems also occur when evaluating the bounds of the next section.

8.3. Bounds for the Inverse

In this section we describe bounds for the inverse of a triangular matrix and show how they can be used to bound condition numbers. All the bounds in this section have the property that they depend only on the absolute values of the elements of the matrix. The norm estimation methods of Chapter 14, on the other hand, do take account of the signs of the elements.

The bounds are all based on the following theorem, whose easy proof we omit.

Theorem 8.11. *If U is a nonsingular upper triangular matrix then*

$$|U^{-1}| \le M(U)^{-1} \le W(U)^{-1} \le Z(U)^{-1},$$

where the upper triangular matrices $W(U)$ and $Z(U)$ are defined as follows:

$$w_{ij} = \begin{cases} |u_{ii}|, & i = j, \\ -\max_{i+1\le k\le n} |u_{ik}|, & i < j, \end{cases}$$

$$z_{ij} = \begin{cases} \alpha, & i = j, \\ -\alpha\beta, & i < j, \end{cases}$$

where $a = \min_k |u_{kk}|$, $b = \max_{i<j} |u_{ij}|/|u_{ii}|$. $\square$

Theorem 8.11 is a special case of results in the theory of Ill-matrices. For more general results couched in terms of matrix minorants and diagonal dominance, respectively, see Dahlquist [261, 1983] and Varga [1051, 1976]; see also Householder [587, 1964, Exercise 15, p. 58].

An obvious implication of the theorem is that for any vector z and any absolute norm

$$\| |U^{-1}||z| \| \le \| M(U)^{-1}|z| \| \le \| W(U)^{-1}|z| \| \le \| Z(U)^{-1}|z| \|.$$

By taking $z = |U|e$, $z = |U||x|$, and $z = e$, respectively, we obtain upper bounds for cond(U), cond(U,x), and $\kappa_\infty(U)$. The cost of computing these bounds is just the cost of solving a triangular system with coefficient matrix $M(U)$, $W(U)$, or $Z(U)$, which is easily seen to be $O(n^2)$. $O(n)$, and $O(1)$ flops, respectively. By comparison, computing any of these condition numbers exactly costs $O(n^3)$ flops.

As an example, here is how to compute an upper bound for $\|T^{-1}\|_\infty$ in n^2 flops.

Algorithm 8.12. Given a nonsingular upper triangular matrix $U \in \mathbb{R}^{n \times n}$, this algorithm computes $m = \|M(U)^{-1}\|_\infty \ge \|U^{-1}\|_\infty$.

```
$y_n = 1/|u_{nn}|$
for $i = n - 1:-1:1$
    $s = 1$
    for $j = i + 1:n$
        $s = s + |u_{ij}|y_j$
    end
    $y_i = y_i/|u_{ii}|$
end
$m = \|y\|_\infty$
```

How good are these upper bounds? We know from Problem 8.2 that the ratio $\|M(T)^{-1}\|/\|T^{-1}\|$ can be arbitrarily large, therefore any of the upper bounds can be arbitrarily poor. However, with suitable assumptions on T, more can be said.

It is easy to show that if T is bidiagonal then $|T^{-1}| = M(T)^{-1}$. Since a bidiagonal system can be solved in $O(n)$ flops, it follows that the three condition numbers of interest can each be computed exactly in $O(n)$ flops when T is bidiagonal.

As in the previous section, triangular matrices that result from a pivoting strategy also lead to a special result.

Theorem 8.13. *Suppose the upper triangular matrix* $U \in \mathbb{R}^{n\times n}$ *satisfies*

$$|u_{ii}| \geq |u_{ij}| \text{ for all } j > i.$$

Then, for the 1-, 2-, *and* ∞*--norms,*

$$\frac{1}{\min_i |u_{ii}|} \leq \|U^{-1}\| \leq \|M(U)^{-1}\| \leq \|W(U)^{-1}\| \leq \|Z(U)^{-1}\| \leq \frac{2^{n-1}}{\min_i |u_{ii}|}. \tag{8.8}$$

Proof. The left-hand inequality is trivial. The right-hand inequality follows from the expression $\|Z(U)^{-1}\|_{1,\infty} = (\beta+1)^{n-1}/\alpha$ (see Problem 8.5), together with $\|A\|_2 < \sqrt{\|A\|_1\|A\|_\infty}$. □

The inequalities from the second on in (8.8) are all equalities for the matrix with $u_{ii} = 1$ and $u_{ij} = -1$ $(j > i)$. The question arises of whether equality is possible for the upper triangular matrices arising from QR factorization with column pivoting, which satisfy the inequalities (see Problem 18.5)

$$u_{kk}^2 \geq \sum_{i=k}^{j} u_{ij}^2, \qquad j = k+1\colon n, \quad k = 1\colon n. \tag{8.9}$$

That equality is possible is shown by the parametrized matrix of Kahan [626, 1966]

$$U_n(\theta) = \mathrm{diag}(1, s, \ldots, s^{n-1}) \begin{bmatrix} 1 & -c & -c & \ldots & -c \\ & 1 & -c & \ldots & -c \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & -c \\ & & & & 1 \end{bmatrix}, \tag{8.10}$$

where $c = \cos(q)$, $s = \sin(q)$. It is easily verified that $U_n(q)$ satisfies the inequalities (8.9)-as equalities, in fact. From (8.3), $U_n(q)^{-1} = (b_{ij})$ is given by

$$\beta_{ij} = \begin{cases} s^{1-j}, & i = j, \\ s^{1-j}c(c+1)^{j-i-1}, & i < j. \end{cases}$$

Thus as $\theta \to 0$, $s^{n-1}U_n(\theta)^{-1} \to [0, 0, \ldots, 0, x]$, where $= [2^{n-2}, 2^{n-1}, \ldots, 1, 1]^T$, and hence, for small q,

$$\|U_n(\theta)^{-1}\|_{1,2,\infty} \approx \frac{2^{n-1}}{|u_{nn}|}.$$

It can also be verified that the matrix $\overline{U}_n(\theta) = (\overline{u}_{ij})$ defined by $\overline{u}_{ij} = (-1)^{j-i}|u_{ij}|$ satisfies, for small q, $\|\overline{U}_n(\theta)\|^{-1} \approx 1/|u_{nn}|$, while $|M(\overline{U}_n(\theta)\|)^{-1} \approx 2^{n-1}/|u_{nn}|$. Hence the upper bounds for $\|U^{-1}\|$ can all be too big by a factor of order 2^{n-1}.

8.4. A Parallel Fan-In Algorithm

Substitution is not the only way to solve a triangular system. In this section we describe a different approach that has been suggested for parallel computation.

Any lower triangular matrix $L \in \mathbb{R}^{n\times n}$ can be factorized $L = L_1 L_2 \ldots L_n$, where L_k differs from the identity matrix only in the kth column:

$$L_k = \begin{bmatrix} I_{k-1} & & & & \\ & l_{kk} & & & \\ & l_{k+1,k} & 1 & & \\ & \vdots & & \ddots & \\ & l_{nk} & & & 1 \end{bmatrix}. \qquad (8.11)$$

The solution to a linear system $Lx = b$ may therefore be expressed as

$$x = L^{-1}b = M_n M_{n-1} \ldots M_1 b, \qquad (8.12)$$

where $M_i = L_i^{-1}$. When evaluated in the natural right-to-left order, this formula yields a trivial variation of a column-oriented version of substitution.

The fan-in algorithm evaluates the product (8.12) in [log(n + 1)] steps by the fan-in operation (which is the operation used in pairwise summation: see §4.1). For example, for $n = 7$ the calculation is specified by

$$x = \bigl((M_7M_6)(M_5M_4)\bigr)\bigl((M_3M_2)(M_1b)\bigr),$$

where all the products appearing within a particular size of parenthesis can be evaluated in parallel. In general. the evaluation can be expressed as a binary tree of depth [log(n + 1)] + 1, with products M_1b and M_iM_{i-1} (i = 3, 5,. . . , 2[(n - 1)/2] + 1) at the top level and a single product yielding x at the bottom level. This algorithm was proposed and analysed by Sameh and Brent [889, 1977], who show that it can be implemented in $\frac{1}{2}\log^2 n + O(\log n)$ time steps on $\frac{1}{68}n^3 + O(n^2)$ processors. The algorithm requires about $n^3/10$ operations and thus is of no interest for serial computation. Some pertinent comments on the practical significance of log n terms in complexity results are given by Edelman [341, 1993].

To derive an error bound while avoiding complicated notation that obscures the simplicity of the analysis, we take $n = 7$. The result we obtain is easily seen to be valid for all n. We will not be concerned with the precise values of constants, so we write c_n for a constant depending on n. We assume that the inverses $M_i = L_i^{-1}$ are formed exactly, because the errors in forming them affect only the constants. From the error analysis of matrix-vector and matrix-matrix multiplication (§3.5), we find that the computed solution $\widehat{x}$ satisfies

$$\widehat{x} = \bigl((M_7M_6 + \Delta_{76})(M_5M_4 + \Delta_{54}) + \Delta_{7654}\bigr)\bigl((M_3M_2 + \Delta_{32})(M_1 + \Delta_1)b\bigr), \qquad (8.13)$$

where

$$\begin{aligned}
|\Delta_{i,i-1}| &\le c_n u|M_i||M_{i-1}| + O(u^2), \quad i = 5, 7,\\
|\Delta_{7654}| &\le c_n u(|M_7M_6||M_5M_4| + |M_7M_6M_5M_4|) + O(u^2),\\
|\Delta_{32}| &\le c_n u(|M_3||M_2| + |M_3M_2|) + O(u^2),\\
|\Delta_1| &\le c_n u|M_1| + O(u^2).
\end{aligned}$$

Premultiplying (8.13) on the left by L, we find that the residual $r = L\widehat{x} - b$ is a sum of terms of the form

$$L(M_7 \dots M_{j+1})\Delta_{j,\dots,k}M_{k-1}\dots M_1 b = L_1 \dots L_j \Delta_{j,\dots,k} L_k \dots L_7 x.$$

All these terms share the same upper bound, which we derive for just one of them. For $j = 5$, $k =$

$$\begin{aligned}
|L_1 \dots L_5 \Delta_{54} L_4 \dots L_7 x| &\le c_n u|L_1 \dots L_5||M_5||M_4||L_4 \dots L_7 x| + O(u^2)\\
&= c_n u|L_1 \dots L_5||L_6L_7L^{-1}L_1 \dots L_4|\\
&\quad \times |L_5L_6L_7L^{-1}L_1L_2L_3||L_4 \dots L_7 x| + O(u^2)\\
&\le c_n u|L||L^{-1}||L||L^{-1}||L||x| + O(u^2),
\end{aligned}$$

where we have used the property that, for any $L \in \mathbb{R}^{n\times n}$, $|L_1|\dots|L_n| = |L|$. The overall residual bound is therefore of the form

$$|b - L\widehat{x}| \le d_n u|L||L^{-1}||L||L^{-1}||L||x| + O(u^2), \tag{8.14}$$

or, on taking norms,

$$\|b - L\widehat{x}\|_\infty \le d_n u\| \, |L||L^{-1}||L||L^{-1}||L||x| \, \|_\infty + O(u^2). \tag{8.15}$$

By considering the binary tree associated with the fan-in algorithm, and using the fact that the matrices at the ith level of the tree have at most 2^{i-1} nontrivial columns, it is easy to see that we can take d_n = an log n, where a is a constant of order 1.

It is not hard to find numerical examples where the bound in (8.15) is approximately attained (for $d_n = 1$) and greatly exceeds $u\|L\|_\infty\|\widehat{x}\|_\infty$, which is the magnitude required for normwise backward stability. One way to construct such examples is to use direct search (see Chapter 24).

The key fact revealed by (8.15) is that the fan-in algorithm is only conditionally stable. In particular, the algorithm is normwise backward stable if L is well conditioned. A special case in which (8.14) simplifies is when L is an M-matrix and $b \ge 0$: Problem 8.4 shows that in this case $|L^{-1}||L||x| \le (2n-1)|x|$, so (8.14) yields $|L\widehat{x} - b| \le (2n-1)^2 d_n u|L||x| + O(u^2)$, and we have componentwise backward stability (to first order).

We can obtain from (8.15) the result

$$(L+\Delta L)\widehat{x}=b, \qquad \|\Delta L\|_\infty \le \alpha_n u \kappa_\infty(L)^2\|L\|_\infty + O(u^2), \tag{8.16}$$

which was proved by Sameh and Brent (889, 1977] (with $a_n = \frac{1}{4}n^2 \log n + O(n \log n)$). However, (8.16) is a much weaker bound than (8.14) and (8.15). In particular, a diagonal scaling $Lx = b \rightarrow D_1LD_2 \cdot D_2^{-1}x = D_1b$ (where D_j is diagonal) leaves (8.14) (and, to a lesser extent, (8.15)) essentially unchanged, but can change the bound (8.16) by an arbitrary amount.

A forward error bound can be obtained directly from (8.13). We find that

$$\begin{aligned}|x-\widehat{x}| &\le d'_n u|M_7||M_6|\dots|M_1||b| + O(u^2)\\ &= d'_n u M(L)^{-1}|b| + O(u^2),\end{aligned} \tag{8.17}$$

where $M(L)$ is the comparison matrix (a bound of the same form as that in Theorem 8.9 for substitution-see the Notes and References and Problem 8.10). which can be weakened to

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \le c_n u \frac{\|\,M(L)^{-1}|L||x|\,\|_\infty}{\|x\|_\infty} + O(u^2). \tag{8.18}$$

We also have the bound

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \le d_n u \frac{\|\,(|L^{-1}||L|)^3|x|\,\|_\infty}{\|x\|_\infty} + O(u^2), \tag{8.19}$$

which is an immediate consequence of (8.14). Either bound in (8.18) and (8.19) can be arbitrarily larger than the other, for fixed n. An example where (8.19) is the better bound (for large n) is provided by the matrix with $l_{ij} \equiv 1$, for which $|L^{-1}||L|$ has maximum element 2 and $M(L)^{-1}|L|$ has maximum element 2^{n-1}.

8.5. Notes and References

Section 8.2 is based on Higham [538, 1989]. Many of the results presented in §§8.2 and 8.3 have their origin in the work of Wilkinson. Indeed, these sections are effectively a unification and extension of Wilkinson's results in [1085, 1961], [1088, 1963], [1089, 1965].

Classic references for Theorems 8.3 and 8.5 are Wilkinson [1085, 1961, p. 294], [1088, 1963, pp. 100-102], Forsythe and Moler [396, 1967, §21], and Stewart [941, 1973, pp. 150, 408-410].

Analogues of Theorem 8.7 and Corollary 8.10 for matrix inversion are proved by Wilkinson in [1085, 1961, pp. 322 323], and Corollary 8.10 itself is proved in [1089, 1965, pp. 250-251].

A result of the form of Theorem 8.9 holds for any triangular system solver that does not rely on algebraic cancellation-in particular, for the fan-in algorithm, as already seen in (8.17). See Problem 8.10 for a more precise formulation of this general result.

The bounds in §8.3 have been investigated by various authors. The unified presentation given here is based on Higham [534, 1987]. Karasalo [642, 1974] derives an $O(n)$ flops algorithm for computing $\|M(T)^{-1}\|_F$. Manteuffel [726, 1981] derives the first two inequalities in Theorem 8.11, and Algorithm 8.12. A different derivation of the equations in Algorithm 8.12 is given by Jennings [613, 1982, §9]. The formulae given in Problem 8.5 are derived directly as upper bounds for $\|T^{-1}\|_{1,\infty}$ by Lemeire [699, 1975].

That $\|B^{-1}\|_\infty$ can be computed in $O(n)$ flops when B is bidiagonal, as $\|M(B)^{-1}e\|_\infty$, was first pointed out by Higham [531, 1986]. Demmel and Kahan [296, 1990] derive an estimate for the smallest singular value $\sigma_{\min}$ of a bidiagonal matrix B by using the inequality $\overline{\sigma} \le \sigma_{\min}(B) \le \sqrt{n}\overline{\sigma}$, where $\overline{\sigma} = \min(\|B^{-1}\|_\infty^{-1}, \|B^{-1}\|_1^{-1})$ They compute $\overline{\sigma}$ in $O(n)$ flops as

$$\overline{\sigma} = \min(\|M(B)^{-1}e\|_\infty^{-1}, \|M(B)^{-T}e\|_\infty^{-1})$$

Section 8.4 is adapted from Higham [560, 1995], in which error analysis is given for several parallel methods for solving triangular systems.

The fan-in method is topical because the fan-in operation is a special case of the parallel prefix operation and several fundamental computations in linear algebra are amenable to a parallel prefix-based implementation, as discussed by Demmel [287, 1992], [288, 1993]. (For a particularly clear explanation of the parallel prefix operation see the textbook by Buchanan and Turner [154, 1992, §13.21.) The important open question of the stability of the parallel prefix implementation of Sturm sequence evaluation for the symmetric tridiagonal eigenproblem has recently been answered by Mathias [734, 1995]. Mathias shows that for positive definite matrices the relative error in a computed minor can be as large as a multiple of λ_n^{-3}, where λ_n is the smallest eigenvalue of the matrix; the corresponding bound for serial evaluation involves λ_n^{-1}. The analogy with (8.19), where we also see a condition cubing effect, is intriguing.

Higham and Pothen [568, 1994] analyse the stability of the "partitioned inverse method" for parallel solution of sparse triangular systems with many right-hand sides. This method has been studied by several authors in the 1990s; see Alvarado, Pothen, and Schreiber [13, 1993] and the references therein. The idea of the method is to factor a sparse triangular matrix $L \in \mathbb{R}^{n\times n}$ as $L = L_1L_2 \dots L_n = G_1G_2 \dots G_m$, where each G_i is a product of consecutive L_j terms and $1 \le m \le n$, with m as small as possible subject to the G_i being sparse. Then the solution to $Lx = b$ is evaluated as

$$x = G_m^{-1}G_{m-1}^{-1}\dots G_1^{-1}b,$$

where each G_i^{-1} is formed explicitly and the product is evaluated from right to left. The advantage of this approach is that x can be computed in m serial steps of parallel matrix-vector multiplication.

8.5.1. LAPACK

Computational routine xTRTRS solves a triangular system with multiple right-hand sides: xTBTRS is an analogue for banded triangular matrices. There is no driver routine for triangular systems.

Problems

> *Before you start an exercise session,*
> *make sure you have a glass of water and*
> *a mat or towel nearby.*
> -MARIE HELVIN, *Model Tips for a Healthy Future (1994)*

8.1. Show that under the no-guard-digit model (2.6). Lemma 8.2 remains true if (8.1) is changed to

$$b_k\widehat{y}(1+\theta_k) = c - \sum_{i=1}^{k-1} a_i b_i (1+\theta_{i+2}),$$

and that the corresponding modification of Theorem 8.5 has

$$(T+\Delta T)\widehat{x} = b, \qquad |\Delta T| \le \gamma_{n+1}|T|.$$

8.2. Show that for a triangular matrix T the ratio $\|M(T)^{-1}\|/\|T^{-1}\|$ can be arbitrarily large.

8.3. Suppose the unit upper triangular matrix $U \in \mathbb{R}^{n\times n}$ satisfies $|u_{ij}| \le 1$ for $j > i$. By using Theorem 8.9. show that the computed solution $\widehat{x}$ from substitution on $Ux = b$ satisfies

$$|x_i - \widehat{x}_i| \le 2^{n-i}\bigl((n^2+n+1)u + O(u^2)\bigr)\|b\|_\infty.$$

Compare with the result of applying Theorem 8.7.

8.4. Let $T \in \mathbb{R}^{n\times n}$ be triangular and suppose $T = M(T)$ and $Tx = b > 0$. *Show* that $|T^{-1}||T||x| \le (2n-1)|x|$, and hence that cond$(T, x) \le 2n-1$. This result show that a triangular M-matrix system with a nonnegative right-hand side is very well conditioned with respect to componentwise relative perturbations, irrespective of the size of $\kappa(T)$ (and so leads to an alternative proof of Corollary 8.10).

8.5. Show that for a triangular $T \in \mathbb{R}^{n\times n}$, $\|Z(T)^{-1}\| = (\beta+1)^{n-1}/\alpha$ for both the 1- and ∞-norms (a and b are defined in Theorem 8.11).

8.6. Write detailed algorithms for efficiently computing $\| M(U)^{-1}|z| \|_\infty$ and $\| W(U)^{-1}|z| \|_\infty$.

8.7. Bounds from diagonal dominance. (a) Prove the following result (Ahlberg and Nilson [8, 1963], Varah [1049, 1975]): if $A \in \mathbb{R}^{n\times n}$ (not necessarily triangular) satisfies

$$\alpha_i = |a_{ii}| - \sum_{j\neq i} |a_{ij}| > 0, \qquad i = 1\colon n$$

(that is, A is strictly diagonally dominant by rows), then

$$\|A^{-1}\|_\infty \le \frac{1}{\min_i \alpha_i}.$$

(b) Hence show that (Varga [1051, 1976]) if $A \in \mathbb{R}^{n\times n}$ satisfies

$$\beta_i = |a_{ii}|d_i - \sum_{j\neq i} |a_{ij}|d_j > 0, \qquad i = 1\colon n,$$

for some positive diagonal matrix $D = \mathrm{diag}(d_i)$ (that is, AD is strictly diagonally dominant by rows), then

$$\|A^{-1}\|_\infty \le \frac{\|D\|_\infty}{\min_i \beta_i}.$$

(c) Use part (b) to provide another derivation of the upper bound $\|M(T)^{-1}e\|_\infty \ge \|T^{-1}\|_\infty$.

8.8. (a) Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. For a given i and j, determine, if possible, α_{ij} such that $A + \alpha_{ij}e_ie_j^T$ is singular. Where is the "best" place to perturb A to make it singular?

(b) Let $T = U(1)$ in (8.2), so that, for example,

$$T_4 = \begin{bmatrix} 1 & -1 & -1 & -1 \\ & 1 & -1 & -1 \\ & & 1 & -1 \\ & & & 1 \end{bmatrix}.$$

Show that T_n is made singular by subtracting 2^{2-n} from a certain element of T_n.

8.9. (Zha [1127, 1993]) Show that if c and s are nonnegative (with $c^2 + s^2 = 1$) then the Kahan matrix $U_n(q)$ in (8.10) has $s^{n-2}\sqrt{1+c}$ as its second smallest singular value. (That there should be such an explicit formula is surprising; none is known for the smallest singular value.)

8.10. Consider a method for solving triangular systems $Tx = b$ that computes $x_i = f_i(T, b)$ where, for all i, f_i is a multivariate rational function in which the only divisions are by diagonal elements of L and such that when $T = M(T)$ and $b \geq 0$ there are no subtractions in the evaluation of f_i. Show that a bound holds of the form in Theorem 8.9, namely, for $T \in \mathbb{R}^{n\times n}$,

$$|x - \widehat{x}| \leq \bigl(c_n u + O(u^2)\bigr) M(T)^{-1}|b|. \tag{8.20}$$

Give an example of a triangular system solver for which (8.20) is not satisfied.

Chapter 9
LU Factorization and Linear Equations

It appears that Gauss and Doolittle applied
the method only to symmetric equations.
More recent authors, for example, Aitken, Banachiewicz, Dwyer, and Crout . . .
have emphasized the use of the method, or variations of it,
in connection with non-symmetric problems . . .
Banachiewicz . . . saw the point . . .
that the basic problem is really one of matrix factorization,
or "decomposition" as he called it.

-PAUL S. DWYER, *Linear Computations (1951)*

Intolerable pivot-growth [with partial pivoting] is a phenomenon that happens
only to numerical analysts who are looking for that phenomenon.

-WILLIAM M. KAHAN, *Numerical Linear Algebra* (1966)

By 1949 *the major components of the*
Pilot ACE were complete and undergoing trials . . .
During 1951 *a programme for solving simultaneous*
linear algebraic equations was used for the first time.
26th June, 1951 *was a landmark in the history of the machine,*
for on that day it first rivalled alternative computing methods
by yielding by 3 p.m. the solution to
a set of 17 *equations submitted the same morning.*

-MICHAEL WOODGER, *The History and Present Use of*
Digital Computers at the National Physical Laboratory (1958).

The closer one looks,
the more subtle and remarkable Gaussian elimination appears.

-LLOYD N. TREFETHEN, *Three Mysteries of Gaussian Elimination* **(1985)**

9.1. Gaussian Elimination

We begin by giving a traditional description of Gaussian elimination (GE) for solving a linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular.

The strategy of GE is to reduce a problem we can't solve (a full linear system) to one that we can (a triangular system), using elementary row operations. There are $n - 1$ stages, beginning with $A^{(1)} := A$, $b^{(1)} := b$, and finishing with the upper triangular system $A^{(n)}x = b^{(n)}$.

At the kth stage we have converted the original system to $A^{(k)}x = b^{(k)}$, where

$$A^{(k)} = \begin{bmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{bmatrix},$$

with $A_{11}^{(k)} \in \mathbb{R}^{(k-1)\times(k-1)}$ upper triangular. The purpose of the kth stage of the elimination is to zero the elements below the diagonal in the kth column of $A^{(k)}$. This is accomplished by the operations

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

where the multipliers $m_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}$, $i = k+1\colon n$. At the end of the $(n - 1)$st stage we have the upper triangular system $A^{(n)}x = b^{(n)}$, which is solved by back substitution. For an $n \times n$ matrix, GE requires $2n^3/3$ flops.

There are two problems with the method as described. First. there is a breakdown with division by zero if $a_{kk}^{(k)} = 0$. Second, if we are working in finite precision and some multiplier m_{ik} is large, then there is a possible loss of significance: in the subtraction $a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)}$, low-order digits of $a_{ij}^{(k)}$ could be lost. Losing these digits could correspond to making a relatively large change to the original matrix A. The simplest example of this phenomenon is for the matrix $\left[\begin{smallmatrix} \epsilon & 1 \\ 1 & 1 \end{smallmatrix}\right]$; here. $a_{22}^{(2)} = 1 - 1/\epsilon$, and $fl(a_{22}^{(2)}) = -1/\epsilon$ if $\epsilon < u$, which would be the exact answer if we changed a_{22} from 1 to 0.

These observations motivate the strategy of *partial pivoting*. At the start of the kth stage, the kth and rth rows are interchanged, where

$$|a_{rk}^{(k)}| := \max_{k\le i\le n} |a_{ik}^{(k)}|.$$

Partial pivoting ensures that the multipliers are nicely bounded:

$$|m_{ik}| \le 1, \quad i = k+1\colon n.$$

A more expensive pivoting strategy, which interchanges both rows and columns, is *complete pivoting*.

At the start of the kth stage, rows k and r and columns k and s are interchanged, where

$$|a_{rs}^{(k)}| := \max_{k\le i,j\le n} |a_{ij}^{(k)}|.$$

Note that this requires $O(n^3)$ comparisons in total, compared with $O(n^2)$ for partial pivoting. Because of the searching overhead, and because partial pivoting works so well, complete pivoting is rarely used in practice.

Much insight into GE is obtained by expressing it in matrix notation. We can write

$$A^{(k+1)} = M_k A^{(k)} := \begin{bmatrix} I_{k-1} & & & & \\ & 1 & & & \\ & -m_{k+1,k} & 1 & & \\ & -m_{k+2,k} & & \ddots & \\ & \vdots & & & \ddots & \\ & -m_{n,k} & & & & 1 \end{bmatrix} A^{(k)}.$$

The matrix M_k can be expressed compactly as $M_k = I - m_k e_k^T$, where e_k is the kth unit vector and $e_i^T m_k = 0$ for $i \le k$. To invert M_k, just flip the signs of the multipliers: $M_k^{-1} = I + m_k e_k^T$. Overall,

$$M_{n-1}M_{n-2}\dots M_1 A = A^{(n)} =: U,$$

and so

$$\begin{aligned} A &= M_1^{-1}M_2^{-1}\dots M_{n-1}^{-1}U \\ &= (I + m_1 e_1^T)(I + m_2 e_2^T)\dots(I + m_{n-1}e_{n-1}^T)U \\ &= \left(I + \sum_{i=1}^{n-1} m_i e_i^T\right)U \\ &= \begin{bmatrix} 1 & & & & \\ m_{21} & 1 & & & \\ \vdots & m_{32} & \ddots & & \\ \vdots & \vdots & & \ddots & \\ m_{n1} & m_{n2} & \dots & m_{n,n-1} & 1 \end{bmatrix} U =: LU. \end{aligned}$$

The conclusion is that GE computes an LU factorization of A: $A = LU$, where L is unit lower triangular and U is upper triangular.

We introduce the shorthand notation $A_k := A(1{:}k,\ 1{:}k)$.

Theorem 9.1. *There exists a unique LU factorization of* $A \in \mathbb{R}^{n\times n}$ *if and only if* A_k *is nonsingular for* $k = 1{:}n - 1$. *If* A_k *is singular for some* $1 \le k \le n - 1$ *then the factorization may exist, but if so it is not unique.*

Proof. Suppose A_k is nonsingular for $k = 1{:}n - 1$. The existence of an LU factorization can be proved by. examining the steps of GE. but a more elegant proof, which also gives uniqueness, can be obtained by an inductive bordering construction. Suppose A_{k-1} has the unique LU factorization $A_{k-1} = L_{k-1}U_{k-1}$ (this supposition clearly holds for $k - 1 = 1$). We look for a factorization

$$A_k = \begin{bmatrix} A_{k-1} & b \\ c^T & a_{kk} \end{bmatrix} = \begin{bmatrix} L_{k-1} & 0 \\ l^T & 1 \end{bmatrix} \begin{bmatrix} U_{k-1} & u \\ 0 & u_{kk} \end{bmatrix} =: L_kU_k.$$

The equations to be satisfied are $L_{k-1}u = b$, $U_{k-1}^T l = c$, and $a_{kk} = l^Tu + u_{kk}$. The matrices L_{k-1} and U_{k-1} are nonsingular, since $0 \neq \det(A_{k-1}) = \det(L_{k-1})\det(U_{k-1})$. so the equations have a unique solution. completing the induction.

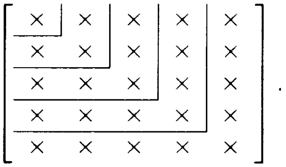
We prove the converse, under the assumption that A is nonsingular; for the case A singular see Problem 9.1. Suppose an LU factorization exists. Then $A_k = L_kU_k$ for $k = 1{:}n$, which gives

$$\det(A_k) = \det(U_k) = u_{11} \dots u_{kk}. \tag{9.1}$$

Setting $k = n$ we find that $0 \neq \det(A) = u_{11} \dots u_{nn}$, and hence $\det(A_k) = u_{11} \dots u_{kk} \neq 0$, $k = 1{:}n - 1$.

Examples that illustrate the last part of the theorem are $\begin{bmatrix}0&0\\0&0\end{bmatrix} = \begin{bmatrix}1&0\\l&1\end{bmatrix}\begin{bmatrix}0&0\\0&0\end{bmatrix}$, which holds for any l, and $\begin{bmatrix}0&1\\1&1\end{bmatrix}$ which does not have an LU factorization. □

Visually, the condition of Theorem 9.1 is (for $n = 5$) that the indicated submatrices must be nonsingular:



From (9.1) follows the expression $u_{kk} = \det(A_k)/\det(A_{k-1})$. In fact, all the elements of L and U can be expressed by determinantal formulae (see, e.g., Gantmacher [413, 1959, p. 35] or Householder [587, 1964. p. 11]):

$$l_{ij} = \frac{\det\big(A([1{:}j-1,\, i], 1{:}j)\big)}{\det(A_j)}, \quad i \geq j, \tag{9.2a}$$

$$u_{ij} = \frac{\det\big(A(1{:}i, [1{:}i-1,\, j])\big)}{\det(A_{i-1})}, \quad i \leq j. \tag{9.2b}$$

The effect of partial pivoting is easily seen by considering the case $n = 4$. We have

$$\begin{aligned} U &= M_3P_3M_2P_2M_1P_1A, \quad \text{where } P_k \text{ swaps rows } k, r\ (r \geq k), \\ &= M_3 \cdot \underbrace{P_3M_2P_3}_{} \cdot \underbrace{P_3P_2M_1P_2P_3}_{} \cdot \underbrace{P_3P_2P_1A}_{} \\ &=: M_3'M_2'M_1'PA, \end{aligned}$$

where, for example, $M_1' = P_3P_2(I - m_1e_1^T)P_2P_3 = I - (P_3P_2m_1)e_1^T$. For $k = 1,2,3$, M_k' is the same as M_k except the multipliers are interchanged. Hence, for $n = 4$, GE with partial pivoting (GEPP) applied to A is equivalent to GE without pivoting applied to the row-permuted matrix PA. This conclusion is true for any n: GEPP computes a factorization $PA = LU$. Similarly, GE with complete pivoting computes a factorization $PAQ = LU$, where P and Q are permutation matrices.

Exploitation of the LU factorization streamlines both the error analysis and the practical solution of linear systems. Solution of $Ax = b$ breaks into a factorization phase, $PA = LU$ for partial pivoting $(O(n^3)$ flops), and a substitution phase, where the triangular systems $Ly = Pb$, $Ux = y$ are solved $(O(n^2)$ flops). If more than one system is to be solved with the same coefficient matrix but different right-hand sides, the factorization can be reused, with a consequent saving in work.

Computing an LU factorization $A = LU$ is equivalent to solving the equations

$$a_{ij} = \sum_{r=1}^{\min(i,j)} l_{ir}u_{rj}.$$

If these nonlinear equations are examined in the right order, they are easily solved. For added generality let $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ and consider an LU factorization with $L \in \mathbb{R}^{m\times n}$ and $U \in \mathbb{R}^{n\times n}$ (L is lower trapezoidal: $l_{ij} = 0$ for $i < j$). Suppose we know the first $k - 1$ columns of L and the first $k - 1$ rows of U. Setting $l_{kk} = 1$,

$$a_{kj} = l_{k1}u_{1j} + \cdots + l_{k,k-1}u_{k-1,j} + \boxed{u_{kj}}, \quad j = k\colon n, \tag{9.3}$$

$$a_{ik} = l_{i1}u_{1k} + \cdots + \boxed{l_{ik}}\,u_{kk}, \quad i = k + 1\colon m. \tag{9.4}$$

We can solve for the boxed elements in the kth row of U and then the kth column of L. This process is called Doolittle's method.

Algorithm 9.2 (Doolittle's method). This algorithm computes an LU factorization $A = LU \in \mathbb{R}^{m\times n}$, where $m \geq n$ (assuming the factorization exists), by Doolittle's method.

```
for k = 1:n
    for j = k:n
(*)     $u_{kj} = a_{kj} - \sum_{i=1}^{k-1} l_{ki}u_{ij}$
    end
    for i = k + 1:m
(**)    $l_{ik} = (a_{ik} - \sum_{j=1}^{k-1} l_{ij}u_{jk})/u_{kk}$
    end
end
```

Cost: $n^2(m - n/3)$ flops.

Doolittle's met hod is mathematically equivalent to GE without pivoting, for we have, in (9.3).

$$a_{kj} - l_{k1}u_{1j} - \cdots - l_{ks}u_{sj} \equiv a_{kj}^{(s+1)} \quad (j > k), \tag{9.5}$$

and similarly for (9.3). Had we chosen the normalization $u_{ii} \equiv 1$, we would have obtained the Crout method. The Crout and Doolittle methods are well suited to calculations by hand or with a desk calculator, because they obviate the need to store the intermediate quantities $a_{ij}^{(k)}$. They are also attractive when we can accumulate inner products in extended precision.

It is straightforward to incorporate partial pivoting into Doolittle's method (see. e.g.. Stewart [941, 1973, p. 138]). However, complete pivoting cannot be incorporated without changing the met hod.

9.2. Error Analysis

The error analysis of GE is a combination of the error analyses of inner products and substitution. When this fact is realized. the analysis becomes straight forward. The key observation leading to this viewpoint is that all mathematically equivalent variants of GE satisfy a common error bound. To see why, first note the connection between standard GE, as we have described it, and the Doolittle met hod, as shown in (9.5). Whether the inner product in (9.5) is calculated as one operation. or whether its terms are calculated many operations apart, precisely the same rounding errors are sustained (assuming that extended precision accumulation of inner products is not used): all that changes is the moment when those rounding errors are committed. If we allow inner products to be reordered. so that, for example, the summation (*) in Algorithm 9.2 is calculated with the index i decreasing from k - 1 to 1, instead of increasing from 1 to k - 1. then the actual rounding errors are different but a common bound holds for all orderings.

It suffices, then, to analyse the Doolittle method. It also suffices to analyse the met hod without pivoting, because GE with partial or complete pivoting is equivalent to GE without pivoting applied to a permuted matrix.

The assignments (*) and (**) in Algorithm 9.2 are of the form $y = (c - \sum_{i=1}^{k-1} a_i b_i)/b_k$, which is analysed in Lemma 8.4. Applying the lemma, we deduce that, no matter what the-ordering of the inner products, the computed matrices L and U satisfy (with $l_{kk} := 1$)

$$\Big|a_{kj} - \sum_{i=1}^{k-1} \widehat{l}_{ki}\widehat{u}_{ij} - \widehat{u}_{kj}\Big| \le \gamma_k \sum_{i=1}^{k} |\widehat{l}_{ki}||\widehat{u}_{ij}|, \quad j \ge k,$$
$$\Big|a_{ik} - \sum_{j=1}^{k} \widehat{l}_{ij}\widehat{u}_{jk}\Big| \le \gamma_k \sum_{j=1}^{k} |\widehat{l}_{ij}||\widehat{u}_{jk}|, \quad i > k.$$

These inequalities constitute a backward error result for LU factorization.

Theorem 9.3. *If GE applied to $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$ runs to completion then the computed LU factors $\widehat{L} \in \mathbb{R}^{m\times n}$ and $\widehat{U} \in \mathbb{R}^{n\times n}$ satisfy*

$$\widehat{L}\widehat{U} = A + \Delta A, \qquad |\Delta A| \le \gamma_n |\widehat{L}||\widehat{U}|. \qquad \square \tag{9.6}$$

With only a little more effort, a backward error result can be obtained for the solution of $Ax = b$.

Theorem 9.4. *Let $A \in \mathbb{R}^{n\times n}$ and suppose GE produces computed LU factors $\widehat{L}$, $\widehat{U}$, and a computed solution $\widehat{x}$ to $Ax = b$. Then*

(9.7)

Proof. From Theorem 9.3, $\widehat{L}\widehat{U} = A + \Delta A_1$, $|\Delta A_1| \le \gamma_n|\widehat{L}||\widehat{U}|$. By Theorem 8.5, substitution produces $\widehat{y}$ and $\widehat{x}$ satisfying

$$(\widehat{L} + \Delta L)\widehat{y} = b, \qquad |\Delta L| \le \gamma_n|\widehat{L}|,$$
$$(\widehat{U} + \Delta U)\widehat{x} = \widehat{y}, \qquad |\Delta U| \le \gamma_n|\widehat{U}|.$$

Thus

$$b = (\widehat{L}+\Delta L)(\widehat{U}+\Delta U)\widehat{x} = (A+\Delta A_1+\widehat{L}\Delta U+\Delta L\widehat{U}+\Delta L\Delta U)\widehat{x} = (A+\Delta A)\widehat{x},$$

where $|\Delta A| \le 3\gamma_n|\widehat{L}||\widehat{U}| + \gamma_n^2|\widehat{L}||\widehat{U}|$. We need a constant $2\gamma_n$ instead of $3\gamma_n+\gamma_n^2$. Although it is not usually worth expending effort reducing constants in error bounds (see the Wilkinson quotation at the start of Chapter 10), we will optimize constants in this important case. Consideration of Lemma 8.4 shows that we actually have

$$|\Delta L| \le \mathrm{diag}(\gamma_{i-1})|\widehat{L}|, \qquad |\Delta U| \le \mathrm{diag}(\gamma_{n-i+1})|\widehat{U}|,$$

so that

$$\begin{aligned}|\widehat L\Delta U + \Delta L\widehat U + \Delta L\Delta U| &\le \operatorname{diag}(\gamma_{i-1} + \gamma_{n-i+1} + \gamma_{i-1}\gamma_{n-i+1})|\widehat L||\widehat U| \\ &\le \gamma_n|\widehat L||\widehat U|,\end{aligned}$$

using Lemma 3.3, which gives the required constant. □

How do we interpret Theorem 9.4? Ideally, we would like $|\Delta A| \le u|A|$, which corresponds to the uncertainty introduced by rounding the elements of A, but because each element of A undergoes up to n arithmetic operations we cannot expect better than a bound $|\Delta A| \le c_n u|A|$, where c_n is a constant of order n. Such a bound holds if $\widehat L$ and $\widehat U$ satisfy $|L||U| = |LU|$, which certainly holds if $\widehat L$ and $\widehat U$ are nonnegative, because then (9.6) gives

$$\begin{aligned}|\widehat L||\widehat U| = |\widehat L\widehat U| = |A + \Delta A| &\le |A| + \gamma_n|\widehat L||\widehat U| \\ \Rightarrow |\widehat L||\widehat U| &\le \frac{1}{1-\gamma_n}|A|.\end{aligned} \tag{9.8}$$

Substituting into (9.7), we obtain

$$(A + \Delta A)\widehat x = b, \qquad |\Delta A| \le \frac{2\gamma_n}{1-\gamma_n}|A| \qquad (\widehat L, \widehat U \ge 0).$$

This result says that $\widehat x$ has a small componentwise relative backward error.

One class of matrices that has nonnegative LU factors is defined as follows. $A \in \mathbb{R}^{n\times n}$ is *totally positive (nonnegative)* if the determinant of every square submatrix is positive (nonnegative). In particular, this definition requires that a_{ij} and $\det(A)$ be positive or nonnegative. Some examples of totally nonnegative matrices are given in Chapter 26. If A is totally nonnegative then it has an LU factorization $A = LU$ in which L and U are totally nonnegative, so that $L \ge 0$ and $U \ge 0$ (see Problem 9.6); moreover, the computed factors $\widehat L$ and $\widehat U$ are nonnegative for sufficiently small values of the unit roundoff u [273, 1977]. Inverses of totally nonnegative matrices also have the property that $|A| = |L||U|$ (see Problem 9.7). Note that the property of a matrix or its inverse being totally nonnegative is generally destroyed under row permutations. Hence for totally nonnegative matrices and their inverses it is best to use Gaussian elimination *without* pivoting.

One important fact that follows from (9.6) and (9.7) is that the stability of GE is determined not by the size of the multipliers but by the size of the matrix $|\widehat L||\widehat U|$. This matrix can be small when the multipliers $\widehat l_{ij}$ are large, and large when the multipliers are of order 1 (as we will see in the next section).

To understand the stability of GE further we turn to norms. For GE without pivoting, the ratio $\|\,|L||U|\,\|/\|A\|$ can be arbitrarily large. For example, for the matrix $\left[\begin{smallmatrix}\epsilon & 1\\ 1 & 1\end{smallmatrix}\right]$ the ratio is of order ϵ^{-1}. Assume then that partial pivoting is used. Then $|l_{ij}| \le 1$ for all $i \ge j$, since the l_{ij} are the multipliers.

And it is easy to show by induction that $|u_{ij}| \leq 2^{i-1} \max_{k\leq i} |a_{kj}|$. Hence, for partial pivoting, L is small and U is bounded relative to A.

Traditionally, backward error analysis for GE is expressed in terms of the *growth factor*

$$\rho_n = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|},$$

which involves all the elements $a_{ij}^{(k)}$ $(k = 1{:}n)$ that occur during the elimination. Using the bound $|u_{ij}| = |a_{ij}^{(i)}| \leq \rho_n \max_{i,j}|a_{ij}|$ we obtain the following classic theorem.

Theorem 9.5 (Wilkinson). *Let* $A \in \mathbb{R}^{n\times n}$ *and suppose GE with partial pivoting produces a computed solution* $\widehat{x}$ *to* $Ax = b$. *Then*

$$(A + \Delta A)\widehat{x} = b, \qquad \|\Delta A\|_\infty \leq 2n^2\gamma_n\rho_n\|A\|_\infty. \qquad \square \tag{9.9}$$

We hasten to admit to using an illicit manoeuvre in the derivation of this theorem: we have used bounds for L and U that strictly are valid only for the exact L and U. We could instead have defined the growth factor in terms of the computed $\widehat{a}_{ij}^{(k)}$, but then any bounds for the growth factor would involve the unit roundoff (similarly, we can only guarantee that $|\widehat{l}_{ij}| \leq 1 + u$). Our breach of correctness is harmless for the purposes to which we will put the theorem.

The assumption in Theorem 9.5 that partial pivoting is used is not necessary: essentially the same result holds for GE without pivoting (see Problem 9.8). The normwise backward stability of GE with or without pivoting is therefore governed by the growth factor, to which we now turn our attention.

9.3. The Growth Factor

It is easy to show that $\rho_n \leq 2^{n-1}$ for partial pivoting. Wilkinson notes that this upper bound is achieved for matrices of the form illustrated for $n = 4$ by

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix}.$$

For these matrices, no interchanges are required by partial pivoting, and there is exponential growth of elements in the last column of the reduced matrices. In fact, this is just one of a nontrivial class of matrices for which partial pivoting achieves maximal growth. When necessary in the rest of this chapter, we denote the growth factor for partial pivoting by ρ_n^p and that for complete pivoting by ρ_n^c.

Theorem 9.6 (Higham and Higham). *All real* $n \times n$ *matrices* A *for which* $\rho_n^p(A) = 2^{n-1}$ *are of the form*

$$A = DM \begin{bmatrix} T & \vdots & \alpha d \\ 0 & \vdots & \end{bmatrix},$$

where $D = \mathrm{diag}(\pm 1)$. *M is unit lower triangular with* $m_{ij} = -1$ *for* $i > j$, *T is an arbitrary nonsingular upper triangular matrix of order* $n-1$, $d = (1,2,4,,.... 2^{n-1})^T$, *and a is a scalar such that* $a := |a_{1n}| = \max_{i,j} |a_{ij}|$.

Proof. GE with partial pivoting applied to a matrix A gives a factorization $B := PA = LU$, where P is a permutation matrix. It is easy to show that $|u_{ij}| < 2^{i-1}\, max_{r\le i}\, |b_{rj}|$, with equality for $i = s$ only if there is equality for $i = 1:s-1$. Thus $p_n = 2^{n-1}$ implies that the last column of U has the form aDd, and also that $|b_{1n}| = \max_{i,j} |b_{ij}|$. By considering the final column of B, and imposing the requirement that $|l_{ij}| \le 1$, it is easy to show that the unit lower triangular matrix L must have the form $L = DMD$. It follows that at each stage of the reduction every multiplier is ±1; hence no interchanges are performed, that is, $P = I$. The only requirement on T is that it be nonsingular, for if $t_{ii} = 0$ then the ith elimination stage would be skipped because of a zero pivot column and no growth would be produced on that stage. □

Note that by varying the elements m_{ij} $(i > j)$ and the vector d in Theorem 9.6 we can construct matrices for which ρ_n^p achieves any desired value between 1 and 2^{n-1}.

Despite the exist once of matrices for which p_n is large with partial pivoting, the growth factor is almost invariably small in practice. For example, Wilkinson says "It is our experience that any substantial increase in size of elements of successive A_r is extremely uncommon even with partial pivoting No example which has arisen naturally has in my experience given an increase by a factor as large as 16" [1089, 1965, pp. 213-214].

Until recently. there were no reports in the literature of large growth factors being observed in practical applications. However, Wright [1116, 1993] has found a class of two-point boundary value problems that. when solved by the multiple shooting met hod, yield a linear system for which partial pivoting suffers exponential growth. The matrix is block lower bidiagonal, except for a nonzero block in the top right-hand corner. Furthermore, Foster [399, 1994] shows that a quadrature met hod for solving a practically occurring Volterra integral equation gives rise to linear systems for which partial pivoting again gives large growth factors.

There exist some well-known matrices that give unusually large, but not exponential, growth. They can be found using the following theorem, which is applicable whatever the strategy for interchanging rows and columns in GE.

Theorem 9.7 (Higham and Higham). *Let $A \in \mathbb{C}^{n\times n}$ be nonsingular and set $a = \max_{i,j} |a_{ij}|$, $b = \max_{i,j} |(A^{-1})_{ij}|$, and $q = (ab)^{-1}$. Then $q \le n$, and for any permutation matrices P and Q such that PAQ has an LU factorization. the growth factor p_n for GE without pivoting on PAQ satisfies $p_n \ge q$.*

Proof. The inequality $q \le n$ follows from $\sum_{j=1}^{n} a_{ij}(A^{-1})_{ji} = 1$. Consider an LU factorization $PAQ = LU$ computed by GE. We have

$$\begin{aligned} |u_{nn}^{-1}| &= |e_n^T U^{-1} e_n| = |e_n^T U^{-1} L^{-1} e_n| = |e_n^T Q^T A^{-1} P^T e_n| \\ &= |(A^{-1})_{ij}| \quad \text{for some } i,\, j \\ &\le \beta. \end{aligned} \tag{9.10}$$

Hence $\max_{i,j,k} |a_{ij}^{(k)}| \ge |u_{nn}| \ge \beta^{-1}$, and the result follows.

Note that $q^{-1} = ab$ satisfies $\kappa_\infty(A)^{-1} \le \theta^{-1} \le n^2 \kappa_\infty(A)^{-1}$. Clearly, A has to be very well conditioned for the theorem to provide a lower bound q near the maximum of n.

We apply the theorem to three noncontrived matrices that appear in practical applications.

(1) The matrix

$$S_n = \sqrt{\frac{2}{n+1}}\left(\sin\left(\frac{ij\pi}{n+1}\right)\right)_{i,j=1}^{n}, \tag{9.11}$$

is the symmetric, orthogonal eigenvector matrix for the second difference matrix (the tridiagonal matrix with typical row (-1, 2, -1)-see §26.5); it arises, for example, in the analysis of time series [19, 1971, §6.5]. Theorem 9.7 gives $p_n(S_N) \ge (n+1)/2$.

(2) A Hadamard matrix H_n is an $n \times n$ matrix with elements $h_{ij} = \pm 1$ and for which the rows of H_n are mutually orthogonal. Hadamard matrices exist only for certain n; a necessary condition for their existence if $n > 2$ is that n is a multiple of 4. For more about Hadamard matrices see Hall [494, 1967, Chap. 14], Wallis [1062, 1993], and Wallis, Street, and Wallis [1063, 1972]. We have $H_n H_n^T = nI$, and so $H_n^{-1} = n^{-1} H_n^T$. Theorem 9.7 gives $p_n > n$.

(3) The next matrix is a complex Vandermonde matrix based on the roots of unity, which occurs in the evaluation of Fourier transforms (see §23.1):

$$V_n = \Big(\exp\big(-2\pi i(r-1)(s-1)/n\big)\Big)_{r,s=1}^{n}. \tag{9.12}$$

Since $V_n^{-1} = n^{-1}V_n^H$, Theorem 9.7 gives $p_n(V_n) \geq n$.

Note that each of these matrices is orthogonal or unitary (to within a row scaling in the case of the Hadamard matrix), so it is not necessary to apply GE to them! This may explain why growth factors of order n for these matrices have not been reported in the literature as occurring in practice.

To summarize, although there are practically occurring matrices for which partial pivoting yields a moderately large, or even exponentially large, growth factor, the growth factor is almost invariably found to be small. Explaining this fact remains one of the major unsolved problems in numerical analysis. The best attempt to date is by Trefethen and Schreiber [1019, 1990]. who develop a statistical model of the average growth factor for partial pivoting and complete pivoting. Their model supports their empirical findings that for various distributions of random matrices the average growth factor (normalized by the standard deviation of the initial matrix elements) is close to $n^{2/3}$ for partial pivoting and $n^{1/2}$ for complete pivoting. Extensive experiments by Edelman suggest that for random matrices from the normal $N(0, 1)$ distribution the unnormalized growth factor for partial pivoting grows like $n^{1/2}$ [345, 1995].

We turn now to complete pivoting. Wilkinson [1085, 1961, pp. 282-285] showed that

$$\rho_n^c \leq n^{1/2}(2\cdot 3^{1/2}\cdots n^{1/(n-1)})^{1/2} \sim c\,n^{1/2}n^{\frac{1}{4}\log n}, \tag{9.13}$$

and that this bound is not attainable. The bound is a much more slowly growing function than 2^{n-1}, but can still be quite large (e.g., it is 3570 for $n = 100$). As for partial pivoting, in practice the growth factor is usually small. Wilkinson stated that "no matrix has been encountered in practice for which p_1/p_n was as large as 8" [1085, 1961, p. 285] and that "no matrix has yet been discovered for which $f(r) > r$" [1089, 1965, p. 213] (here. p_i is the $(n - i + 1)$st pivot and $f(r) := \rho_r^c$).

Cryer [256, 1968] defined

$$g(n) = \sup_{A\in\mathbb{R}^{n\times n}} \rho_n^c(A). \tag{9.14}$$

The following results are known:

- $g(2) = 2$ (trivial).
- $g(3) = 2\frac{1}{4}$; Tornheim [1012, 1965] and Cohen [229, 1974].
- $g(4) = 4$; Cryer [256, 1968] and Cohen [229, 1974].
- $g(5) < 5.005$; Cohen [229, 1974].

Tornheim [1012, 1965] (see also Cryer [256, 1968]) showed that $\rho_n^c(H_n) \geq n$ for any $n \times n$ Hadamard matrix H_n (a bound which, as we saw above, holds for any form of pivoting). For n such that a Hadamard matrix does not exist, the best known lower bound is $g(n) \geq \rho_n^c(S_n) = (n+1)/2$ (see (9.11)).

Cryer [256, 1968] conjectured that for real matrices $\rho_n^c(A) \leq n$, with equality if and only if A is a Hadamard matrix. The conjecture $\rho_n^c(A) \leq n$ became one of the most famous open problems in numerical analysis, and has been investigated by many mathematicians. The conjecture was finally shown to be false in 1991. Using a package LANCELOT [236, 1992] designed for large-scale nonlinear optimization, Gould [474, 1991] discovered a 13 × 13 matrix for which the growth factor is 13.0205 in IEEE double precision floating point arithmetic. Edelman subsequently showed, using the symbolic manipulation packages Mathematica and Maple, that a growth factor 13.02 can be achieved in exact arithmetic by making a small perturbation (of relative size 10^{-7}) to one element of Gould's matrix [338, 1992], [348, 1991]. A more striking counterexample to the conjecture is a matrix of order 25 for which $\rho_{25}^c = 32.986341$ [338, 1992]. Interesting problems remain, such as determining $\lim_{n\to\infty} g(n)/n$ and evaluating ρ_n^c for Hadamard matrices (see Problem 9.15).

For complex matrices the maximum growth factor is at least n for any n, since $\rho_n^c(V_n) \geq n$ (see (9.12)). The growth can exceed n, even for $n = 3$: Tornheim [1012, 1965] constructed the example

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & z & z^{-1} \\ 1 & z^{-1} & z \end{bmatrix}, \qquad z = (-1 + i\sqrt{8})/3,$$

for which $\rho_3^c(A) = 3.079$.

9.4. Special Matrices

For matrices with certain special properties, more can be said about the behaviour of GE and, in particular, the size of the growth factor.

As a first example, suppose $A \in \mathbb{C}^{n\times n}$ is diagonally dominant by rows,

$$\sum_{j\neq i} |a_{ij}| \leq |a_{ii}|, \quad i = 1\colon n,$$

or diagonally dominant by columns, that is, A^* is diagonally dominant by rows. Then GE without pivoting is perfectly stable.

Theorem 9.8 (Wilkinson). *If $A \in \mathbb{C}^{n\times n}$ is diagonally dominant by rows or columns then A has an LU factorization without pivoting and the growth factor $\rho_n \leq 2$. If A is diagonally dominant by columns then $|l_{ij}| \leq 1$ for all i and j in the LU factorization without pivoting (hence GEPP does not require any row interchanges).*

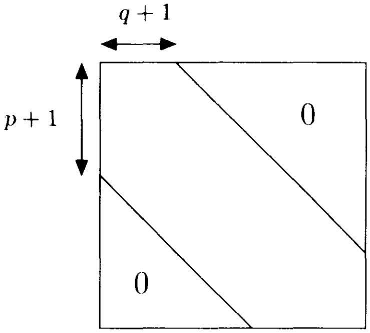


Figure 9.1. *A banded matrix.*

Proof. The result follows immediately from the more general Theorems 12.5 and 12.6 for block diagonally dominant matrices. □

Note that for a matrix diagonally dominant by rows the multipliers can be arbitrarily large but, nevertheless, $p_n \le 2$, so GE is perfectly stable.

A smaller bound for the growth factor also holds for an upper Hessenberg matrix. (H is upper Hessenberg if $h_{ij} = 0$ for $i > j + 1$.)

Theorem 9.9 (Wilkinson). *If $A \in \mathbb{C}^{n\times n}$ is upper Hessenberg then $\rho_n^p \le n$.*

Proof. The structure of an upper Hessenberg H means that at each stage of GEPP we just add a multiple of the pivot row to the next row (after possibly swapping these two rows). That $\rho_n^p \le n$ is a consequence of the following claim, which is easily proved by induction: at the start of stage k, row $k+1$ of the reduced matrix is the same as row $k+1$ of the original matrix, and the pivot row has elements of modulus at most k times the largest element of H. □

A matrix $A \in \mathbb{C}^{n\times n}$ has *lower bandwidth* p and *upper bandwidth* q if $a_{ij} = 0$ for $i > j + p$ and $j > i + q$; see Figure 9.1. It is well known that in an LU factorization of a banded matrix the factors inherit A's band structure: L has lower bandwidth p and U has upper bandwidth q. If partial pivoting is used then, in $PA = LU$. L has at most $p + 1$ nonzeros per column and U has upper bandwidth $p + q$. (For proofs of these properties see Golub and Van Loan [470, 1989, §4.3].) It is not hard to see that for a banded matrix, g_n in Theorem 9.3 can be replaced by $\gamma_{\max(p+1,\,q+1)}$ and $2g_n$ in Theorem 9.4 can be replaced by $\gamma_{\max(p+1,\,q+1)} + \gamma_{p+q+1}$.

The following result bounds the growth factor for partial pivoting on a banded matrix.

Theorem 9.10 (Bohte). *If* $A \in \mathbb{C}^{n\times n}$ *has upper and lower bandwidths p then* $\rho_n^p \le 2^{2p-1} - (p-1)2^{p-2}$, *and this bound is almost attainable when* $n = 2p+1$.

Proof. See Bohte [131, 1975]. An example with $n = 9$ and $p = 4$ in which equality is almost attained is the matrix

$$A = \begin{bmatrix} -1 & -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1+\epsilon & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & -1 & 1 & 0 & 0 & 1 \\ 0 & 0 & -1 & -1 & -1 & -1 & 1 & 0 & 1 \\ 0 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 1 \end{bmatrix},$$

where ϵ is an arbitrarily small positive number, which ensures that rows 1 and 5 are interchanged on the first stage of the elimination, this being the only row interchange required. Ignoring terms in ϵ, the last column of U in $PA = LU$ is $[1, 1, 2, 4, 8, 16, 31, 60, 116]^T$ and the growth factor is 116. □

A special case of Theorem 9.10 is the easily verified result that for a tridiagonal matrix, $\rho_n^p \le 2$. Hence GEPP achieves a small normwise backward error for tridiagonal matrices. In the next section we show that for several types of tridiagonal matrix GE *without pivoting* achieves a small *componentwise* backward error.

9.5. Tridiagonal Matrices

Consider the nonsingular tridiagonal matrix

$$A = \begin{bmatrix} d_1 & e_1 & & & \\ c_2 & d_2 & e_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & e_{n-1} \\ & & & c_n & d_n \end{bmatrix} \in$$

and assume that A has an LU factorization $A = LU$, where

$$L = \begin{bmatrix} 1 & & & & \\ l_2 & 1 & & & \\ & l_3 & 1 & & \\ & & \ddots & \ddots & \\ & & & l_n & 1 \end{bmatrix}, \quad U = \begin{bmatrix} u_1 & e_1 & & & \\ & u_2 & e_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & e_{n-1} \\ & & & & u_n \end{bmatrix}. \qquad (9.15)$$

GE for computing L and U is described by the recurrence relations

$$u_1 = d_1; \qquad \left.\begin{aligned} l_i &= c_i/u_{i-1} \\ u_i &= d_i - l_i e_{i-1} \end{aligned}\right\} \; i = 2\colon n. \tag{9.16}$$

For the computed quantities, we have

$$\begin{aligned} (1+\epsilon_i)\widehat{l}_i &= \frac{c_i}{\widehat{u}_{i-1}}, \qquad |\epsilon_i| \le u, \\ (1+\theta_i)\widehat{u}_i &= d_i - \widehat{l}_i e_{i-1}(1+\delta_i), \qquad |\theta_i|, |\delta_i| \le u. \end{aligned}$$

Hence

$$\begin{aligned} |c_i - \widehat{l}_i\widehat{u}_{i-1}| &\le u|\widehat{l}_i\widehat{u}_{i-1}|, \\ |d_i - \widehat{l}_i e_{i-1} - \widehat{u}_i| &\le u\big(|\widehat{l}_i e_{i-1}| + |\widehat{u}_i|\big). \end{aligned}$$

In matrix terms these bounds may be written as

$$A = \widehat{L}\widehat{U} + \Delta A, \qquad |\Delta A| \le u|\widehat{L}||\widehat{U}|. \tag{9.17}$$

If the LU factorization is used to solve a system $Ax = b$ by forward and back substitution then it is straightforward to show that the computed solution $\widehat{x}$ satisfies

$$(\widehat{L} + \Delta L)(\widehat{U} + \Delta U)\widehat{x} = b, \quad |\Delta L| \le u|\widehat{L}|, \quad |\Delta U| \le (2u + u^2)|\widehat{U}|. \tag{9.18}$$

Combining (9.17) and (9.18) we have, overall,

$$(A + \Delta A)\widehat{x} = b, \qquad |\Delta A| \le f(u)|\widehat{L}||\widehat{U}|, \quad f(u) = 4u + 3u^2 + u^3. \tag{9.19}$$

The backward error result (9.19) applies to arbitrary nonsingular tridiagonal A having an LU factorization. We are interested in determining classes of tridiagonal A for which a bound of the form $|\Delta A| \le g(u)\,|A|$ holds. Such a bound will hold if $|\widehat{L}||\widehat{U}| = |\widehat{L}\widehat{U}|$, as noted in §9.2 (see (9.8)).

Three classes of matrices for which $|\widehat{L}||\widehat{U}| = |\widehat{L}\widehat{U}|$ holds for the exact L and U are identified in the following theorem.

Theorem 9.11. *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular and tridiagonal. If any of the following conditions hold then A has an LU factorization and $|L||U| = |LU|$:*

(a) *A is symmetric positive definite;*

(b) *A is totally nonnegative, or equivalently, $L \ge 0$ and $U \ge 0$;*

(c) *A is an M-matrix, or equivalently, L and U have positive diagonal elements and nonpositive off-diagonal elements;*

(d) *A is sign equivalent to a matrix B of type* (a)-(c), *that is, $A = D_1BD_2$, where $|D_1| = |D_2| = I$.*

Proof. For (a), it is well known that a symmetric positive definite A has an LU factorization in which $U = DL^T$, where D is diagonal with positive diagonal elements. Hence $|L||U| = |L||D||L^T| = |LDL^T| = |LU|$, where the middle equality requires a little thought. In (b) and (c) the equivalences, and the existence of an LU factorization, follow from known results on totally nonnegative matrices [258, 1976] and M-matrices [94, 1994]; $|L||U| = |LU|$ is immediate from the sign properties of L and U. (d) is trivial. □

For diagonally dominant matrices, $|L||U|$ is not equal to $|LU| = |A|$, but it cannot be much bigger.

Theorem 9.12. *Suppose $A \in \mathbb{R}^{n\times n}$ is nonsingular, tridiagonal, and diagonally dominant by rows or columns, and let A have the LU factorization $A = LU$. Then $|L||U| \le 3|A|$.*

Proof. If $|i - j| = 1$ then $(|L||U|)_{ij} = |a_{ij}|$, so it suffices to consider the diagonal elements and show that (using the notation of (9.15))

$$|l_i e_{i-1}| + |u_i| \le 3|d_i|.$$

The rest of the proof is for the case where A is diagonally dominant by rows; the proof for diagonal dominance by columns is similar.

First, we claim that $|e_i| \le |u_i|$ or all i. The proof is by induction. For $i = 1$ the result is immediate, and if it is true for $i - 1$ then, from (9.16).

$$\begin{aligned}|u_i| &\ge |d_i| - |l_i||e_{i-1}| = |d_i| - \frac{|c_i|}{|u_{i-1}|}|e_{i-1}| \\ &\ge |d_i| - |c_i| \ge |e_i|,\end{aligned}$$

as required. Note that, similarly, $|u_i| \le |d_i| + |c_i|$. Finally,

$$\begin{aligned}|l_i e_{i-1}| + |u_i| &= \left|\frac{c_i}{u_{i-1}} e_{i-1}\right| + |u_i| \le |c_i| + |u_i| \\ &\le |c_i| + \big(|d_i| + |c_i|\big) \\ &\le 3|d_i|. \quad \square\end{aligned}$$

Theorem 9.13. *If the nonsingular tridiagonal matrix A is of type* (a)-(d) *in Theorem* 9.11, *and if the unit roundoff u is sufficiently small, then GE for solving $Ax = b$ succeeds and the computed solution $\widehat{x}$ satisfies*

$$(A + \Delta A)\widehat{x} = b, \qquad |\Delta A| \le h(u)|A|, \qquad h(u) = \frac{4u + 3u^2 + u^3}{1 - u}.$$

The same conclusion is true if A is diagonally dominant by rows or columns, with no restriction on u, provided the bound is multiplied by 3.

Proof. If u is sufficiently small then for types (a)-(c) the diagonal elements of U will be positive, since $\widehat{u}_i \to u_i > 0$ as $u \to 0$. It is easy to see that $\widehat{u}_i > 0$ for all i ensures that $|\widehat{L}||\widehat{U}| = |\widehat{L}\widehat{U}|$. The argument is similar for type (d). The result therefore follows from (9.19) and (9.8). The last part is trivial. □

A corollary of Theorem 9.13 is that it is not necessary to pivot for the matrices specified in the theorem (and, indeed, pivoting could vitiate the result of the theorem). Note that large multipliers may occur under the conditions of the theorem, but they do not affect the stability. For example, consider the M-matrix

$$A = \begin{bmatrix} 2 & -2 & 0 \\ \epsilon - 2 & 2 & 0 \\ 0 & -1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ (\epsilon - 2)/2 & 1 & 0 \\ 0 & -1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & 3 \end{bmatrix} = LU,$$

where $0 \le \epsilon < 2$. The multiplier l_{32} is unbounded as $\epsilon \to 0$, but $|L||U| = |A|$ and GE performs very stably, as Theorem 9.13 shows it must.

9.6. Historical Perspective

GE was the first numerical algorithm to be subjected to rounding error analysis, so it is instructive to follow the development of the error analysis from its beginnings in the 1940s.

In the 1940s there were three major papers giving error analyses of GE. Hotelling [583, 1943] presented a short forward error analysis of the LU factorization stage of GE. Under the assumptions that $|a_{ij}| \le 1$ and $|b_i| \le 1$ for all i and j and that the pivots are all of modulus unity, Hotelling derives a bound containing a fact or 4^{n-1} for the error in the elements of the reduced upper triangular system. Hotelling's work was quoted approvingly by Bargmann. Montgomery, and von Newmann [55, 1946], who dismiss elimination met hods for the solution of a linear system $Ax = b$ as being numerically unstable. Instead, they recommended computation of A^{-1} via the Newton Schulz iteration [908, 1933] (which was also discussed by Hotelling). In one paragraph they out line the alleged shortcomings of elimination methods as follows:

> In the elimination method a series of n compound operations is performed each of which depends on the preceding. An error at any stage affects all succeeding results and may become greatly magnified; this explains roughly why instability should be expected. It should be noticed that at each step a division is performed by a number whose size cannot be estimated in advance and which might the so small that any error in it would be greatly magnified by division. In fact such small divisors must occur if the determinant of the matrix is small and may occur even if it is not

> . . . Another reason to expect instability is that once the variable x_n is obtained all the other variables are expressed in terms of it.

As Wilkinson [1098, 1974, p. 354] notes of this paragraph, "almost every statement in it is either wrong or misleading".

Hotelling's result led to general pessimism about the practical effectiveness of GE for solving large systems of equations. Three papers later in the same decade helped to restore confidence in GE.

Goldstine [460, 1972, p. 290] says of his discussions with von Neumann:

> We did not feel it reasonable that so skilled a computer as Gauss would have fallen into the trap that Hotelling thought he had noted . . . Von Neumann remarked one day that even though errors may build up during one part of the computation, it was only relevant to ask how effective is the numerically obtained solution, not how close were some of the auxiliary numbers, calculated on the way to their correct counterparts. We sensed that at least for positive definite matrices the Gaussian procedure could be shown to be quite stable.

von Neumann and Goldstine [1057, 1947] subsequently gave a long and difficult rigorous fixed-point error analysis for the inversion of a symmetric positive definite matrix A via GE. They showed that the computed inverse $\widehat{X}$ satisfies $\|A\widehat{X}-I\|_2 \le 14.2n^2u\kappa_2(A)$. Parlett [821, 1990] explains that "the joy of this result was getting a polynomial in n, and the pain was obtaining 14.2, a number that reflects little more than the exigencies of the analysis." Wilkinson [1095, 1971] gives an interesting critique of von Neumann and Goldstine's paper and points out that the residual bound could hardly be improved using modern error analysis techniques. In a later paper [462, 1951], Goldstine and von Neumann gave a probabilistic analysis, which Goldstine summarizes as showing that "under reasonable probabilistic assumptions the error estimates of the previous paper could be reduced from a proportionality of n^2 to n" [460, 1972, p. 291].

In his 1970 Turing Award Lecture [1096, 1971], Wilkinson recounts how in the early 1940s he solved a system of 12 linear equations on a desk calculator, obtaining a small residual. He goes on to describe a later experience:

> It happened that some time after my arrival [at the National Physical Laboratory in 1946], a system of 18 equations arrived in Mathematics Division and after talking around it for some time we finally decided to abandon theorizing and to solve it . . . The operation was manned by Fox, Goodwin, Turing, and me, and we decided on Gaussian elimination with complete pivoting . . . Again the system was mildly ill-conditioned, the last equation had a coefficient of order 10^{-4} (the original coefficients being of order unity) and

> the residuals were again of order 10^{-10}, that is of the size corresponding to the exact solution rounded to ten decimals. It is interesting that in connection with this example we subsequently performed one or two steps of what would now be called “iterative refinement,” and this convinced us that the first solution had had almost six correct figures.

(Fox [403, 1987] notes that the computation referred to in this quotation took about two weeks using desk computing equipment!) In a subsequent paper, Fox, Huskey, and Wilkinson [404, 1948] presented empirical evidence in support of GE, commenting that “in our practical experience on matrices of orders up to the twentieth, some of them very ill-conditioned, the errors were in fact quite small”.

The experiences of Fox, Huskey, and Wilkinson prompted Turing to write a remarkable paper “Rounding-off errors in matrix processes” [1027, 1948]. In this paper, Turing made several import ant contributions. He formulated the LU (actually. the LDU) factorization of a matrix, proving the “if” part of Theorem 9.1 and showing that GE computes an LDU factorization. He introduced the term “condition number” and defined two matrix condition numbers, one of which is $n^{-1}N(A)N(A^{-1})$, where $N(A) = ||A||_F$, the “N-condition number of A”. He used the word “preconditioning” to mean improving the condition of a system of linear equations (a term that did not come into popular use until the 1970s). He described iterative refinement for linear systems. He exploited backward error ideas, for example by noting that *‘the triangular resolution obtained is an exact resolution of a matrix $A - S$, where $M(S) < \epsilon$” ($M(S) := \max_{i,j} |s_{ij}|$. Finally, and perhaps most importantly, he analysed GEPP for general matrices and obtained a bound for $\|x - \widehat{x}\|_\infty$ that contains a term proportional to $\|A^{-1}\|_\infty^2$. (By making a trivial change in the analysis, namely replacing $A^{-1}b$ by x, Turing’s bound can be made proportional only to $\|A^{-1}\|_\infty$.) Turing also showed that the factor 4^{n-1} in Hotelling’s bound can be improved to 2^{n-1} and that still the bound is attained only in exceptional cases.

In a review of Turing’s paper, Bodewig [129, 1949] described the error bounds as “impractical“ and advocated computing the residual of the computed solution and then determining “the exact correct ion by solving a new system.” That another researcher could miss the point of Turing’s analysis emphasizes how new the concept of rounding error analysis was in the 1940s.

Table 9.1 shows the time for solution of linear systems by GE on some early computing devices. The performance of modern computers on two linear system benchmarks is summarized by Dongarra [312, 1995]; Dongarra’s report is regularly updated and can be obtained from netlib under the benchmark directory.

Douglas [319, 1959] presented a forward error analysis for GE applied to

Table 9.1. *Times for solution of a linear system of order n.*

Machine	Year	n	Time	Reference
Logarithm tables	c. 1884	29 [a]	7 weeks	[952, 1994]
Desk computing equipment	c. 1946	18	2 weeks	[403, 1987]
Harvard Mark 1	1947	10	45 minutes	[b]
IBM 602 Calculating Punch	1949	10	4 hours	[1053, 1949]
Pilot ACE	1951	17	over 3 hours	[1110, 1958]
Pilot ACE’	1954	30	1½ mins	[1110, 1958]
ACE	1958	30	5 seconds	[1110, 1958]
EDSAC 2	1960	31	4 seconds	[73, 1960]
EDSAC 2 [d]	1960	100	7 minutes	[73, 1960]

---

[a] Symmetric positive definite system.

[b] [127, 1948, p. 27], [507, 1948, p. 336].

[c] With magnetic drum store.

[d] Using magnetic tape for auxiliary storage.

diagonally dominant tridiagonal systems arising in the solution of the heat equation by finite differences. He concluded that the whole procedure of solving this partial differential equation “is stable against round-off error”. It is surprising that Douglas’ paper is little known, because irrespective of the fact that his analysis can be simplified and clarified using modern techniques, his is one of the first truly positive rounding error results to be published.

A major breakthrough in the error analysis of GE came with Wilkinson’s pioneering backward error analysis, in which he proved Theorem 9.5 [1085, 1961], [1088, 1963]. Apart from its simplicity and elegance and the realistic nature of the bounds, the main feature that distinguishes Wilkinson’s analysis from the earlier error analyses of GE is that it bounds the normwise backward error rather than the forward error.

Wilkinson had been aware of the properties of the growth factor for partial pivoting long before developing his backward error analysis. In a 1954 paper [1081, 1954] he noted that

> After *m* reductions the largest element is at most 2”’ times as large as the largest original coefficient. It is possible to construct sets in which this factor is achieved but in practice an increase seldom takes place; more frequently the coefficients become progressively smaller, particularly if the equations are ill-conditioned.

This quote summarizes most of what we know today!

Four of the first textbooks to incorporate Wilkinson’s analysis were those of Fox [400, 1964, pp. 161-174], Isaacson and Keller [607, 1966], Wendroff [1074, 1966], and Forsythe and Moler [396, 1967, Chap. 21]. Fox gives a simplified analysis for fixed-point arithmetic under the assumption that the growth factor is of order 1. Forsythe and Moler give a particularly readable backward error analysis that has been widely quoted.

Wilkinson's 1961 result is essentially the best that can be obtained by a normwise analysis. Subsequent work in error analysis for GE has mainly been concerned with bounding the backward error component wise, as in Theorems 9.3 and 9.4. We note that Wilkinson could have given a componentwise bound for the backward perturbation DA, since most of his analysis is at the element level.

Chartres and Geuder [200, 1967] analyse the Doolittle version of GE. They derive a backward error result $(A + DA)\widehat{x} = b$, with a componentwise bound on DA; although they do not recognize it, their bound can be written in the form $|DA| \le c_n u |\widehat{L}||\widehat{U}|$.

Reid [867, 1971] shows that the assumption in Wilkinson's analysis that partial pivoting or complete pivoting is used is unnecessary. Without making any assumptions on the pivoting strategy, he derives for LU factorization the result $LU = A + DA$, $|Da_{ij}| \le 3.01 \min(i - 1, j)\, u \max_k |a_{ij}^{(k)}|$. Again, this is a componentwise bound. Erisman and Reid [355, 1974] note that for a sparse matrix, the term $\min(i - 1, j)$ in Reid's bound can be replaced by m_{ij}, where m_{ij} is the number of multiplications required in the calculation of l_{ij} $(i > j)$ or u_{ij} $(i \le j)$.

de Boor and Pinkus [273, 1977] give the result stated in Theorem 9.4. They refer to the original 1972 German edition of [955, 1980] for a proof of the result and explain several advantages to be gained by working with a componentwise bound for DA, one of which is the strong result that ensues for totally nonnegative matrices. A result very similar to Theorem 9.4 is proved by Sautter [895, 1978].

Skeel [919, 1979] carried out a detailed componentwise error analysis of GE with a different flavour to the analysis given in this chapter. His aim was to understand the numerical stability of GE (in a precisely defined sense) and to determine the proper way to scale a system by examining the behaviour of the backward and forward errors under scaling (see §9.7). He later used this analysis to derive important results about fixed precision iterative refinement (see Chapter 11). Skeel's work popularized the use of component wise backward error analysis and componentwise perturbation theory.

The "$|\widehat{L}||\widehat{U}|$" componentwise style of backward error analysis for GE is now well known, as evidenced by its presence in the text books of Conte and de Boor [237, 1980]. Golub and Van Loan [370, 1989] (also the 1983 first edition), and Stoer and Bulirsch [955, 1980].

Forward error analyses have also been done for GE. The analyses are more complicated and more difficult to interpret than the backward error analyses. Olver and Wilkinson [810, 1982] derive a posteriori forward error bounds that require the computation of A^{-1}. Further results are given in a series of papers by Stummel [965, 1982], [966, 1985], [967, 1985], [968, 1985].

Finally, probabilistic error analysis for GE is given by Barlow and Bareiss [63, 1985].

9.7. Scaling

Prior to solving a linear system $Ax = b$ by GE we are at liberty to scale the rows and the columns:

$$Ax = b \;\rightarrow\; D_1AD_2 \cdot D_2^{-1}x = D_1b, \quad \text{or} \quad A'y = c, \tag{9.20}$$

where D_1 and D_2 are nonsingular diagonal matrices. We apply GE to the scaled system $A'y = c$ and then recover x from $x = D_2y$. Although scaling was used in some of the earliest published programs for GE [396, 1967], [745, 1962], how best to choose a scaling is still not well understood, and no single scaling algorithm can be guaranteed always to perform satisfactorily. Wilkinson's remark "We cannot decide whether equations are ill-conditioned without examining the way in which the coefficients were derived" [1089, 1965, p. 198] sums up the problem of scaling rather well.

The effect of scaling in GE without pivoting is easy to describe. If the elements of D_1 and D_2 are powers of the machine base b (so that the scaling is done without error) and GE produces $\widehat{L}$ and $\widehat{U}$ satisfying $A + DA = \widehat{L}\widehat{U}$, then GE on $A' = D_1AD_2$ produces $D_1\widehat{L}D_1^{-1}$ and $D_1\widehat{U}D_2$ satisfying $A' + D_1DAD_2 = D_1\widehat{L}D_1^{-1} \cdot D_1\widehat{U}D_2$. In other words, the rounding errors in GE scale in the same way as A. This is a result of Bauer [78, 1963] (see [396, 1967, Chap. 11] for a clear proof and discussion). With partial pivoting, however, the choice of pivots is affected by the row scaling (though not the column scaling), and in a way that is difficult to predict.

We can take a method-independent approach to scaling, by considering any method for solving $Ax = b$ that yields a solution satisfying

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \le c_n \kappa_\infty(A)u,$$

with c_n a constant. For the scaled system (9.20) we have

$$\frac{\|D_2^{-1}(x - \widehat{x})\|_\infty}{\|D_2^{-1}x\|_\infty} \le c_n \kappa_\infty(D_1AD_2)u,$$

so it is natural to choose D_1 and D_2 to minimize $\kappa_\infty(D_1AD_2)$. As we saw in §7.3 (Theorem 7.8), the minimum possible value is no larger than $p(|A^{-1}||A|)$. However, a column scaling has the (usually) undesirable effect of changing the norm in which the error is measured. With row scaling only, the minimum value of $\kappa_\infty(D_1A)$ is $\mathrm{cond}(A) = \| |A^{-1}||A| \|_\infty$, achieved when D_1A has rows of unit 1-norm (see (7.12)). Thus row equilibration yields a cond-bounded forward error. For GE, though, it is possible to do even better. Skeel [919, 1979] shows that for $D_1 = \mathrm{diag}(|A||x|)^{-1}$, the forward error bound for GEPP is proportional to $\mathrm{cond}(A,x) = \| |A^{-1}||A||x| \|_\infty / \|x\|_\infty$; the catch is, of course, that the scaling depends on the unknown solution x! Row equilibration can be regarded as approximating x by e in this "optimal" scaling.

The LINPACK LU factorization routines do not include scaling, while in the LAPACK driver routine xGESVX an initial scaling is optional. One reason why scaling is not popular with numerical analysts is that a $\mathrm{cond}(A, x)$-bounded forward error *and* a small componentwise relative backward error are both achieved by fixed precision iterative refinement (assuming it converges): see Chapter 11. Even Skeel's optimal scaling does not guarantee a small componentwise relative backward error.

Some programs for GEPP incorporate row scaling *implicitly*. They compute row scale factors $d_1. . . . , d_n$. but. instead of applying GEPP to $\mathrm{diag}(d_i)^{-1} \times A$, they apply it to A and choose as pivot row at the kth stage a row r for which $d_r|a_{rk}^{(k)}|$ is maximal. This type of scaling has the sole effect of influencing the choice of pivots. There is little justification for using it, and the best bound for the growth factor is 2^{n-1} multiplied by a product of terms d_{i1}/d_{i2} that can be large.

There is, however. one situation in which a form of implicit row scaling is beneficial. Consider the pivoting strategy that selects as the kth pivot an element $a_{rk}^{(k)}$ for which

$$\frac{|a_{rk}^{(k)}|}{\|A^{(k)}(r,k\colon n)\|_\infty} = \max_{i\ge k} \frac{|a_{ik}^{(k)}|}{\|A^{(k)}(i,k\colon n)\|_\infty}. \qquad (9.21)$$

A result of Peña [825, 1994] shows that if there exists a permutation matrix P such that PA has an LU factorization $PA = LU$ with $|PA| = |L||U|$, then such a factorization will be produced by the pivoting scheme (9.21). This means that, unlike for partial pivoting, we can use the pivoting scheme (9.21) with impunity on totally nonnegative matrices and their inverses, row permutations of such matrices, and any matrix for which some row permutation has the "$|PA| = |L||U|$" property. However, this pivoting strategy is as expensive as complete pivoting to implement, and for general A it is not guaranteed to produce a factorization as stable as that produced by partial pivoting.

9.8. A Posteriori Stability Tests

Having solved a linear system by LU factorization we can compute the componentwise or normwise backward error at the cost of evaluating one or two matrix-vector products (see Theorems 7.1 and 7.3). In some situations, though, we may wish to assess the stability of a computed LU factorization before using it to solve one or more linear systems. One possibility is to compute the growth factor by monitoring the size of elements during the elimination, at a cost of $O(n^3)$ comparisons. This has been regarded as rather expensive, and more efficient ways to estimate ρ_n have been sought.

Businger [169, 1971] describes a way to obtain an upper bound for ρ_n in $O(n^2)$ operations. This approach is generalized by Erisman and Reid [355, 1974], who apply the Holder inequality to the equation

$$a_{ij}^{(k+1)} = a_{ij} - \sum_{r=1}^{k} l_{ir}u_{rj}, \qquad i,j > k,$$

to obtain the bound

$$\begin{aligned} |a_{ij}^{(k+1)}| &\le |a_{ij}| + \|(l_{i1},\dots,l_{ik})\|_p \|(u_{1j},\dots,u_{kj})\|_q \\ &\le \max_{i,j} |a_{ij}| + \max_i \|(l_{i1},\dots,l_{i,i-1})\|_p \max_j \|(u_{1j},\dots,u_{j-1,j})\|_q, \end{aligned} \tag{9.22}$$

where $p^{-1} + q^{-1} = 1$. In practice, $p = 1, 2, \infty$ are the values of interest. Barlow [56, 1986] notes that application of the Holder inequality instead to

$$a_{ij}^{(k+1)} = \sum_{r=k+1}^{\min(i,j)} l_{ir}u_{rj}$$

yields a sometimes sharper bound.

It is interesting to note that in light of experience with the bound (9.22), Reid [868, 1987] recommends computing the growth factor explicitly in the context of sparse matrices, arguing that the expense is justified because (9.22) can be a very weak bound. See Erisman et al. [354, 1987] for some empirical results on the quality of the bound.

Chu and George [209, 1985] observe that the ∞-norm of the matrix $|\widehat{L}||\widehat{U}|$ can be computed in $O(n^2)$ operations without forming the matrix explicitly, since

$$\|\,|\widehat{L}||\widehat{U}|\,\|_\infty = \|\,|\widehat{L}||\widehat{U}|e\,\|_\infty = \|\,|\widehat{L}|(|\widehat{U}|e)\,\|_\infty.$$

Thus one can cheaply compute a bound on $\|\Delta A\|_\infty$ from the componentwise backward error bounds in (9.6) and (9.7).

All the methods discussed in this section make use of an a priori error analysis to compute bounds on the backward error. Because the bounds do not take into account the statistical distribution of rounding errors, and because they involve somewhat pessimistic constant terms, they cannot be expected to be very sharp. Thus it is important not to forget that it is straightforward to compute the backward error itself: $A - \widehat{L}\widehat{U}$. Exact computation costs a prohibitively expensive $O(n^3)$ operations, but $\|A - \widehat{L}\widehat{U}\|_1$ can be *estimated* in

$O(n^2)$ operations using the matrix norm estimator in Algorithm 14.4. Another possibility is to use a running error analysis, in which an error bound is computed concurrently with the factors (see 53.3).

9.9. Sensitivity of the LU Factorization

Although Theorem 9.3 bounds the backward error of the computed LU factors L and U, it does not give any indication about the size of the forward errors $L - \widehat{L}$ and $U - \widehat{U}$. For most applications of the LU factorization it is the backward error and not the forward errors that matters, but it is still of some interest to know how big the forward errors can be. This is a question of perturbation theory and is answered by the next result.

Theorem 9.14 (Barrlund, Sun). *Let the nonsingular matrices* $A \in \mathbb{R}^{n\times n}$ *and* $A+DA$ *have LU factorizations* $A = LU$ *and* $A+DA = (L+DL)(U+DU)$, *and assume that* $\|G\|_2 < 1$, *where* $G = L^{-1}DAU^{-1}$. *Then*

$$\max\left\{\frac{\|\Delta L\|_F}{\|L\|_2}, \frac{\|\Delta U\|_F}{\|U\|_2}\right\} \le \frac{\|G\|_F}{1-\|G\|_2} \le \frac{\|L^{-1}\|_2\|U^{-1}\|_2\|A\|_2}{1-\|L^{-1}\|_2\|U^{-1}\|_2\|\Delta A\|_2}\,\frac{\|\Delta A\|_F}{\|A\|_F}. \tag{9.23}$$

Moreover, if $\rho(|\widetilde{G}|) < 1$, *where* $\widetilde{G} = (L + DL)^{-1}DA(U + DU)^{-1}$, *then*

$$|\Delta L| \le |L+\Delta L|\,\mathrm{stril}\big((I-|\widetilde{G}|)^{-1}|\widetilde{G}|\big),$$
$$|\Delta U| \le \mathrm{triu}\big(|\widetilde{G}|(I-|\widetilde{G}|)^{-1}\big)|U+\Delta U|,$$

where stril(·) *and* triu(·) *denote, respectively, the strictly lower triangular part and the upper triangular part of their matrix arguments.* □

The normwise bounds (9.23) imply that $x(A) := \|L^{-1}\|_2\,\|U^{-1}\|_2\,\|A\|_2$ is an upper bound for the condition numbers of L and U under normwise perturbations. We have

$$\kappa_2(A) \le \chi(A) \le \min\{\kappa_2(L), \kappa_2(U)\}\kappa_2(A),$$

and the ratio $x(A)/\kappa_2(A)$ can be arbitrarily large (though if partial pivoting is used then $\kappa_2(L) \le n2^{n-1}$).

The componentwise bounds in Theorem 9.14 are a little unsatisfactory in that they involve the unknown matrices DL and DU, but we can set these terms to zero and obtain a bound correct to first order.

9.10. Notes and References

A variant of GE was used by the Chinese around the first century AD; the *Jiu Zhang Suanshu* (Nine Chapters of the Mathematical Art) contains a worked example for a system of five equations in five unknowns [619, 1991, pp. 156-177]. [696, 1989]

Gauss, who was a great statistician and numerical analyst, developed his elimination method as a tool to help him prove results in linear regression theory. The first published appearance of GE is in his *Theoria Motus* (1809). Stewart [952, 1994] gives a survey of Gauss's work on solving linear systems: see also the afterword in [423, 1995].

The traditional form of GE, as given at the start of this chapter. can be expressed algorithmically as

```
for $k = 1{:}n$
    for $j = k+1{:}n$
        for $i = k + 1{:}n$
            $a_{ij} = a_{ij} - (a_{ik}/a_{kk})a_{kj}$
        end
    end
end
```

This is identified as the *kji* form of GE. Altogether there are six possible orderings of the three loops. Doolittle's method (Algorithm 9.2) is the *ijk* or *jik* variant of Gaussian elimination. The choice of loop ordering does not affect the stability of the computation, but can greatly affect the efficiency of GE on a high performance computer. For more on the different loop orderings of GE see Chapter 12; Dongarra, Gustavson. and Karp [310, 1984]; and the books by Dongarra, Duff, Sorensen, and van der Vorst [315, 1991] and Golub and Van Loan [470, 1989].

This chapter draws on the survey paper Higham [545, 1990]. Theorems 9.6 and 9.7 are from Higham and Higham [562, 1989].

Myrick Hascall Doolittle (1830-1913) was a "computer of the United States coast and geodetic survey" [362, 1987]. Crout's method was published in an engineering journal in 1941 [255, 1941].

GE and its variants were known by various descriptive names in the early days of computing. These include the bordering met hod, the escalator met hod (for matrix inversion), the square root method (Cholesky factorization), and pivotal condensation. A good source for details of these methods is Faddeeva [360, 1959].

In a confidential 1948 report that "covers the general principles of both the design of the [Automatic Computing Engine] and the method of programming adopted for it", Wilkinson gives a program implementing GE with partial pivoting and iterative refinement [1080, 1948, p. 111]. This was probably the first such program to be written and for a machine that had not yet been built!

The terms “partial pivoting“ and “complete pivoting” were introduced by Wilkinson in [1085, 1961]. The pivoting techniques themselves were in use in the 1940s and it is not clear who, if anyone, can be said to have invented them: indeed, von Neumann and Goldstine [1057, 1947, §4.2] refer to complete pivoting as the “customary procedure”.

There is a long history of published programs for GE. beginning with Crout routines of Forsythe [390, 1960], Thacher [999, 1961], McKeeman [745, 1962], and Bowdler, Martin, Peters, and Wilkinson [138, 1966], all written in Algol 60 (which was the “official” language for publishing mathematical software in the 1960s. and a strong competitor to Fort ran for practical use at that time). The GE routines in LAPACK are the latest in a lineage beginning with the Fortran routines decomp and solve in Forsythe and Moler‘s book [396, 1967], and continuing with routines by Moler [766, 1972], [767, 1972] (which achieve improved efficiency in Fortran by accessing arrays by column rather than by row), Forsythe, Malcolm, and Moler [395, 1977] (these routines incorporate condition estimation-see Chapter 14), and LINPACK [307, 1979].

LU factorization of totally nonnegative matrices has been investigated by Cryer [257, 1973], [258, 1976], Ando [21, 1987], and de Boor and Pinkus [273, 1977]. It is natural to ask whether we can test for total nonnegativity without computing all the minors. The answer is yes: for an $n \times n$ matrix total nonnegativity can be tested in $O(n^3)$ operations. as shown by Gasca and Peña [421, 1992]. The test involves carrying out a modified form of GE in which all the elimination operations are between adjacent rows and then checking whether certain pivots are positive. Note the analogy with positive definiteness, which holds for a symmetric matrix if and only if all the pivots in GE are positive.

The dilemma of whether to define the growth factor in terms of exact or computed quantities is faced by all authors; most make one choice or the other, and go on to derive, without comment, bounds that are strictly incorrect. Theorem 9.8, for example, bounds the exact growth factor; the computed one could. conceivably violate the bound, but only by a tiny relative amount. van Veldhuizen [1045, 1977] shows that for a variation of partial pivoting that allows either a row or column interchange at each stage, the growth factor defined in terms of computed quantities is at most about $(1 + 3nu)2^{n-1}$, compared with the bound 2^{n-1} for the exact growth factor.

The idea of deriving error bounds for GE by analysing the equations obtained by solving $A = LU$ is exploited by Wilkinson [1097, 1974], who gives a general analysis that includes Cholesky factorization. This paper gives a concise summary of error analysis of factorization methods for linear equations and least squares problems.

Various authors have tabulated growth factors in extensive tests with random matrices. In tests during the development of LINPACK, the largest value observed was $\rho_{40}^{p} = 23$, occurring for a random matrix of 1s, 0s, and -1s [307, 1979, p. 1.21]. Macleod [720, 1989] recorded a value $\rho_{100}^{p} = 35.1$, which occurred for a symmetric matrix with elements from the uniform distribution on [-1, 1]. In one MATLAB test of 1000 matrices of dimension 100 from the normal $N(0, 1)$ distribution, I found the largest growth factor to be $\rho_{n}^{p} = 9.59$.

Gould [474, 1991] used the optimization LANCELOT [236, 1992] to maximize the nth pivot for complete pivoting as a function of about $n^3/3$ variables comprising the intermediate elements $a_{ij}^{(k)}$ of the elimination; constraints were included that normalize the matrix A, describe the elimination equations, and impose the complete pivoting conditions. Gould's package found many local maxima, and many different starting values had to be tried in order to locate the matrix for which $\rho_{13}^{c} > 13$. In an earlier attempt at maximizing the growth factor, Day and Peterson [271, 1988] used a problem formulation in which the variables are the n^2 elements of A, which makes the constraints and objective function substantially more nonlinear than in Gould's formulation. Using the package NPSOL [444, 1986], they obtained "largest known" growth factors for $5 \leq n \leq 7$.

Theoretical progress into understanding the behaviour of the growth factor for complete pivoting has been made by Day and Peterson [271, 1988], Puschmann and Cortés [849, 1983], Puschmann and Nordio [850, 1985], and Edelman and Mascarenhas [345, 1995].

A novel alternative to partial pivoting for stabilizing GE is proposed by Stewart [942, 1974]. The idea is to modify the pivot element to make it suitably large, and undo this rank one change later using the Sherman-Morrison formula. Stewart gives error analysis that bounds the backward error for this modified form of GE.

Theorem 9.8 is proved for matrices diagonally dominant by columns by Wilkinson [1085, 1961, pp. 288-289]. Theorem 9.9 is proved in the same paper. That $p_n \leq 2$ for matrices diagonally dominant by rows does not appear to be well known, but it is proved by Wendroff [1074, 1966, pp. 122-123]. for example.

The results in §9.5 for tridiagonal matrices are taken from Higham [541, 1990]. Another method for solving tridiagonal systems is cyclic reduction, which was developed in the 1960s [171, 1970]. Error analysis given by Amodio and Mazzia [15, 1994] shows that cyclic reduction is normwise backward stable for diagonally dominant tridiagonal matrices.

The chapter "Scaling Equations and Unknowns" of Forsythe and Moler [396, 1967] is a perceptive, easy to understand treatment that is still well worth reading. Early efforts at matrix scaling for GE were directed to equilibrating either just the rows or the rows and columns simultaneously (so that all the rows and columns have similar norms). An algorithm with the latter aim is described by Curtis and Reid [259, 1972]. Other important references on scaling are the papers by van der Sluis [1040, 1970] and Stewart [945, 1977]. which employ normwise analysis, and those by Skeel [919, 1979], [921, 1981], which use componentwise analysis.

Much is known about the existence and stability of LU factorizations of M-matrices and related matrices. A is an H-matrix if the comparison matrix $M(A)$ (defined in (8.6)) is an M-matrix. Funderlic, Neumann, and Plemmons [410, 1982] prove the existence of an LU factorization for an H-matrix A that is generalized diagonally dominant, that is, DA is diagonally dominant by columns for some nonsingular diagonal matrix D: they show that the growth factor satisfies $\rho_n \le 2 \max_i |d_{ii}|/\min_i |d_{ii}|$. Neumann and Plemmons [791, 1984] obtain a growth factor bound for an inverse of an H-matrix. Ahac, Buoni, and Olesky [7, 1988] describe a novel column-pivoting scheme for which the growth factor can be bounded by n then A is an H-matrix.

The normwise bounds in Theorem 9.14 are due to Barrlund [71, 1991] and the componentwise ones to Sun [972, 1992]. Similar bounds are given by Stewart [951, 1993] and Sun [973, 1992]. Barrlund [72, 1992] describes a general technique for deriving matrix perturbation bounds using integrals.

Interval arithmetic techniques (see §24.4) are worth considering if high accuracy or guaranteed accuracy is required when solving a linear system. We mention just one paper, that by Demmel and Krückeberg [297, 1985], which provides a very readable introduction to the subject and contains further references.

For several years Edelman has been collecting information on the solution of large, dense linear algebra problems. His papers [337, 1991], [341, 1993], [342, 1994] present statistics and details of the applications in which large dense problems arise. Edelman also discusses relevant issues such as what users expect of the computed solutions and how best to make use of parallel computers. Table 9.2 contains "world records" for linear systems from Edelman's surveys. For all the records shown the matrix was complex and the system was solved in double precision arithmetic by some version of LU factorization. Most of the very large systems currently being solved come from the solution of boundary integral equations, a major application being the analysis of radar cross sections; the resulting systems have coefficient matrices that are complex symmetric (but not Hermitian). A recent reference is Wang [1064, 1991].

9.10.1. LAPACK

Driver routines xGESV (simple) and xGESVX (expert) use LU factorization with partial pivoting to solve a general system of linear equations with multiple right-hand sides. The expert driver incorporates iterative refinement, condition estimation, and backward and forward error estimation and has an option to scale the system $AX = B$ to $(D_R^{-1} A D_C^{-1}) D_C X = D_R^{-1} B$ before solution,

Table 9.2. *Records for largest dense linear systems solved (dimension n).*

Year	n	Computer	Time
1991	55,296	Connection Machine CM-2	4.4 days
1992/3	75,264	Intel iPSC/860	$2^{2}/3$ days
1994	76,800	Connection Machine CM-5	4.1 days
1995	128,600	Intel Paragon	≈ 1 hour

where $D_R = \mathrm{diag}(r_i) = \mathrm{diag}(\max_j |a_{ij}|)$ and $D_C = \mathrm{diag}(\max_i r_i|a_{ij}|)$; the scaling is done by the routine xGEEQU. The LU factorization is computed by the routine xGETRF, which uses a partitioned outer product algorithm. The expert driver also returns the quantity $\|A\|/\|U\|$, where $\|A\| := \max_{i,j} |a_{ij}|$, which is an estimate of the reciprocal of the growth factor, $1/\rho_n^p$. A value much less than 1 signals that the condition estimate and forward error bound could be unreliable.

For band matrices, the driver routines are xGBSV and xGBSVX, and for tridiagonal matrices, xGTSV and xGTSVX; again, these use LU factorization with partial pivoting.

Problems

9.1. (Completion of proof of Theorem 9.1.) Show that if a singular matrix $A \in \mathbb{R}^{n\times n}$ has a unique LU factorization then A_k is nonsingular for $k = 1{:}n-1$.

9.2. Define $A(s) = A - sI$, where $\sigma \in \mathbb{R}$ and $A \in \mathbb{R}^{n\times n}$. For how many values of s, at most, does $A(s)$ fail to have an LU factorization without pivoting?

9.3. Show that $A \in \mathbb{R}^{n\times n}$ has a unique LU factorization if 0 does not belong to the field of values $F(A) = \{\, z^*Az/(z^*z) : 0 \neq z \in \mathbb{C}^n \,\}$.

9.4. State analogues of Theorems 9.3 and 9.4 for LU factorization with row and column interchanges: $PAQ = LU$.

9.5. Give a 2×2 matrix A having an LU factorization $A = LU$ such that $|L||U| \le c|A|$ does not hold for any c, yet $\|\,|L||U|\,\|_\infty/\|A\|_\infty$ is of order 1.

9.6. Show that if $A \in \mathbb{R}^{n\times n}$ is nonsingular and totally nonnegative it has an LU factorization $A = LU$ with $L \ge 0$ and $U \ge 0$. (Hint: use the inequality

$$\det(A) \le \det\bigl(A(1{:}p,1{:}p)\bigr)\det\bigl(A(p+1{:}n,p+1{:}n)\bigr), \qquad p = 1{:}n -$$

which holds for any totally nonnegative A [414, 1959, p. 100].) Deduce that the growth factor for GE without pivoting $\rho_n \equiv 1$.

9.7. Show that if $A \in \mathbb{R}^{n\times n}$ is nonsingular and its inverse is totally nonnegative then it has an LU factorization $A = LU$ with $|A| = |L||U|$. (Use the fact that if C is totally nonnegative and nonsingular then $JC^{-1}J$ is totally nonnegative, where $J = \mathrm{diag}((-1)^{i+1})$ (this can be proved using determinantal identities; see [21, 1987, Thm. 3.3]).)

9.8. Show that Theorem 9.5 is valid for GE without pivoting, with a different constant.

9.9. Suppose that GE without pivoting is applied to a linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular, and that all operations are performed exactly except for the division determining a single multiplier l_{ij} (where $i > j$ and $A = LU$), which is computed with relative error ϵ: $l_{ij} = l_{ij}(1+\epsilon)$. Evaluate the difference $x - \widehat{x}$ between the exact and computed solutions. (The answer allows us to see clearly the effect of a computational blunder, which could, for example. be the result of the malfunction of a computer's divide operation.)

9.10. Show that q in Theorem 9.7 satisfies

$$\theta(B) := \theta\left(\begin{bmatrix} A & A \\ A & -A \end{bmatrix}\right) = 2\theta(A).$$

Hence, for g(n) defined in (9.14) and S_n in (9.11). deduce a larger lower bound than $g(2n) \geq \rho_n^c(S_{2n}) = (2n+1)/2$.

9.11. Explain the errors in the following criticism of GE with complete pivoting.

> Gaussian elimination with complete pivoting maximizes the pivot at each stage of the elimination. Since the product of the pivots is the determinant (up to sign), which is fixed, making early pivots large forces later ones to be small. These small pivots will have large relative errors due to the accumulation of rounding errors during the algorithm, and dividing by them therefore introduces larger errors.

9.12. In sparse matrix computations the choice of pivot in GE has to be made with the aim of preserving sparsity as well as maintaining stability. In threshold pivoting, a pivot element is chosen from among those elements in column k that satisfy $|a_{ik}^{(k)}| \geq \tau \max_{m\geq k} |a_{mk}^{(k)}|$, where $\tau \in (0,1]$ is a parameter (see, for example, Duff, Erisman, and Reid [325, 1986, §5.4]). Show that for threshold pivoting

$$\max_i |a_{ij}^{(k)}| \leq (1+\tau^{-1})^{\mu_j} \max_i |a_{ij}|,$$

where m_j is the number of nonzero entries in the jth column of U. Hence obtain a bound for p_n.

9.13. (Research Problem) Obtain sharp bounds for the growth factor for GE with partial pivoting applied to (a) a matrix with lower bandwidth p and upper bandwidth q (thus generalizing Theorem 9.10), and (b) a quasi-tridiagonal matrix (an $n \times n$ matrix that is tridiagonal except for nonzero $(1, n)$ and $(n, 1)$ elements).

9.14. (Research Problem) Explain why the growth factor for GE with partial pivoting is almost always small in practice.

9.15. (Research Problem) For GE with complete pivoting what is the value of $\lim_{n\to\infty} g(n)/n$ (see (9.14))? Is ρ_n^c equal to n for Hadamard matrices?

Chapter 10
Cholesky Factorization

The matrix of that equation system is negative definite-which is a
positive definite system that has been multiplied through by – 1.
For all practical geometries the common finite difference
Laplacian operator gives rise to these,
the best of all possible matrices.
Just about any standard solution method will succeed,
and many theorems are available for your pleasure.
—FORMAN S. ACTON, Numerical Methods That Work (1970)

Many years ago we made out of half a dozen transformers
a simple and rather inaccurate machine for
solving simultaneous equations—the solutions being
represented as flux in the cores of the transformers.
During the course of our experiments we
set the machine to solve the equations—
$X + Y + Z = 1$
$X + Y + Z = 2$
$X + Y + Z = 3$
The machine reacted sharply—it blew the main fuse and put all the lights out.
—B. V. BOWDEN, The Organization of a Typical Machine (1953)

There does seem to be some misunderstanding about the
purpose of an a priori backward error analysis.
All too often, too much attention is paid
to the precise error bound that has been established.
The main purpose of such an analysis is either to
establish the essential numerical stability of an algorithm or to
show why it is unstable and in doing so to
expose what sort of change is necessary to make it stable.
The precise error bound is not of great importance.
—J. H. WILKINSON, Numerics/ Linear Algebra on Digits/ Computers (1974)

10.1. Symmetric Positive Definite

Symmetric positive definiteness is one of the highest accolades to which a matrix can aspire. Symmetry confers major advantages and simplifications in the eigenproblem and, as we will see in this chapter, positive definiteness permits economy and numerical stability in the solution of linear systems. A symmetrical matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $x^TAx > 0$ for all nonzero $x \in \mathbb{R}^n$. Well-known equivalent conditions to $A = A^T$ being positive definite are

- $\det(A_k) > 0$, $k = 1{:}n$, where $A_k = A(1{:}k, 1{:}k)$ is the leading principal submatrix of order k.
- $\lambda_k(A) > 0$, $k = 1{:}n$, where λ_k denotes the kth largest eigenvalue.

The first of these conditions implies that A has an LU factorization, $A = LU$ (see Theorem 9.1). Another characterization of positive definiteness is that the pivots in LU factorization are positive, since $u_{kk} = \det(A_k)/\det(A_{k-1})$. By factoring out the diagonal of U and taking its square root, the LU factorization can be converted into a *Cholesky factorization:* $A = R^TR$, where R is upper triangular with positive diagonal elements. This factorization is so important that it merits a direct proof.

Theorem 10.1. *If $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite then there is a unique upper triangular $R \in \mathbb{R}^{n\times n}$ with positive diagonal elements such that $A = R^TR$.*

Proof. The proof is by induction. The result is clearly true for $n = 1$. Assume it is true for $n - 1$. The leading principal submatrix $A_{n-1} = A(1{:}n{-}1, 1{:}n{-}1)$ is positive definite, so it has a unique Cholesky factorization $A_{n-1} = R_{n-1}^TR_{n-1}$. We have a factorization

$$A = \begin{bmatrix} A_{n-1} & c \\ c^T & \alpha \end{bmatrix} = \begin{bmatrix} R_{n-1}^T & 0 \\ r^T & \beta \end{bmatrix} \begin{bmatrix} R_{n-1} & r \\ 0 & \beta \end{bmatrix} =: R^TR \tag{10.1}$$

if

$$R_{n-1}^Tr = c, \tag{10.2}$$

$$r^Tr + \beta^2 = \alpha. \tag{10.3}$$

Equation (10.2) has a unique solution since R_{n-1} is nonsingular. Then (10.3) gives $b^2 = a - r^Tr$. It remains to check that there is a unique real, positive b satisfying this equation. From the equation

$$0 < \det(A) = \det(R^T)\ \det(R) = \det(R_{n-1})^2 b^2$$

we see that $b^2 > 0$, hence there is a unique $b > 0$. □

The proof of the theorem is constructive, and provides a way to compute the Cholesky factorization that builds R a column at a time. Alternatively, we can work directly from the equations

$$a_{ij} = \sum_{k=1}^{i} r_{ki} r_{kj}, \quad j \geq i,$$

which follow by equating (i, j) elements in $A = R^T R$. By solving these equations in the order (1,1), (1,2), (2,2), (1,3), (2,3), (3,3), . . . , (n,n), we obtain the following algorithm.

Algorithm 10.2. Given a symmetric positive definite $A \in \mathbb{R}^{n\times n}$ this algorithm computes the Cholesky factorization $A = R^T R$.

```
for $j = 1:n$
    for $i = 1:j-1$
        $r_{ij} = (a_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj})/r_{ii}$
    end
    $r_{jj} = (a_{jj} - \sum_{k=1}^{j-1} r_{kj}^2)^{1/2}$
end
```

Cost: $n^3/3$ flops (half the cost of LU factorization).

As for Gaussian elimination (GE), there are different algorithmic forms of Cholesky factorization. Algorithm 10.2 is the *jik* or “sdot” form. We describe the *kij*, outer product form in §10.3.

Given the Cholesky factorization $A = R^T R$, a linear system $Ax = b$ can be solved via the two triangular systems $R^T y = b$ and $Rx = y$.

If we define $D = \mathrm{diag}(r_{ii}^2)$ then the Cholesky factorization $A = R^T R$ can be rewritten as $A = LDL^T$, where $L = R^T \mathrm{diag}(r_{ii})^{-1}$ is unit lower triangular. The LDL^T factorization is sometimes preferred over the Cholesky factorization because it avoids the need to compute the n square roots that determine the r_{ii}. The LDL^T factorization is certainly preferred for solving tridiagonal systems, as it requires n less divisions than Cholesky factorization in the substitution stage. All the results for Cholesky factorization in this chapter have analogues for the LDL^T factorization. Block LDL^T factorization for indefinite matrices is discussed in §10.4.

10.1.1. Error Analysis

Error bounds for Cholesky factorization are derived in a similar way to those for LU factorization. Consider Algorithm 10.2. Using Lemma 8.4 we have

$$\left| a_{ij} - \sum_{k=1}^{i} \widehat{r}_{ki} \widehat{r}_{kj} \right| \leq \gamma_i \sum_{k=1}^{i} |\widehat{r}_{ki}| |\widehat{r}_{kj}|. \tag{10.4}$$

From a variation of Lemma 8.4 in which the division is replaced by a square root (see Problem 10.3), we have

$$\left| a_{jj} - \sum_{k=1}^{j} \widehat{r}_{kj}^{2} \right| \le \gamma_{j+1} \sum_{k=1}^{j} \widehat{r}_{kj}^{2}.$$

A backward error result is immediate.

Theorem 10.3. *If Cholesky factorization applied to the symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$ runs to completion then the computed factor R satisfies*

$$\widehat{R}^T\widehat{R} = A + \Delta A, \qquad |\Delta A| \le \gamma_{n+1}|\widehat{R}^T||\widehat{R}|. \qquad \square \tag{10.5}$$

Theorem 10.4. *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite and suppose Cholesky factorization produces a computed factor R and a computed solution $\widehat{x}$ to Ax = b. Then*

$$(A + \Delta A)\widehat{x} = b, \qquad |\Delta A| \le 2\gamma_{n+1}|\widehat{R}^T||\widehat{R}|. \tag{10.6}$$

Proof. The proof is analogous to the proof of Theorem 9.4. $\square$

These results imply that Cholesky factorization enjoys perfect normwise backward stability. The key inequality is

$$\|\,|R^T||R|\,\|_2 = \|\,|R|\,\|_2^2 \le n\|R\|_2^2 = n\|A\|_2,$$

whose analogue for the computed $\widehat{R}$ is, from (10.5),

$$\|\,|\widehat{R}^T||\widehat{R}|\,\|_2 \le n(1 - n\gamma_{n+1})^{-1}\|A\|_2.$$

Thus (10.6) implies

$$\|\Delta A\|_2 \le \|\,|\Delta A|\,\|_2 \le 2\gamma_{n+1}n(1 - n\gamma_{n+1})^{-1}\|A\|_2 \le 8n(n+1)u\|A\|_2, \tag{10.7}$$

where for the last inequality we have assumed that $n\gamma_{n+1} < 1/2$. Another indicator of stability is that the growth factor for GE is exactly 1 (see Problem 10.4). It is important to realize that the multipliers can be arbitrarily large (consider, for example, $\left[\begin{smallmatrix}\theta^2 & \theta \\ \theta & 2\end{smallmatrix}\right]$ as $\theta \to 0$). But, remarkably, for a positive definite matrix the size of the multipliers has no effect on stability.

Note that the perturbation ΔA in (10.6) is not symmetric, in general, because the backward error matrices for the triangular solves with R and R^T are not the transposes of each other. For conditions guaranteeing that a "small" symmetric ΔA can be found, see Problem 7.11.

The following rewritten version of Theorem 10.3 provides further insight into Cholesky factorization.

Theorem 10.5 (Demmel). *If Cholesky factorization applied to the symmetric positive definite matrix* $A \in \mathbb{R}^{n\times n}$ *runs to completion then the computed factor* $\widehat{R}$ *satisfies*

$$\widehat{R}^T\widehat{R} = A + \Delta A, \qquad |\Delta A| \le (1-\gamma_{n+1})^{-1}\gamma_{n+1}dd^T,$$

where $d_i = a_{ii}^{1/2}$.

Proof. Theorem 10.3 shows that $\widehat{R}^T\widehat{R} = A + \Delta A$ with $|\Delta A| \le \gamma_{n+1}|\widehat{R}^T||\widehat{R}|$. Denoting by $\widehat{r}_i$ the ith column of $\widehat{R}$, we have

$$\|\widehat{r}_i\|_2^2 = \widehat{r}_i^T\widehat{r}_i = a_{ii} + \Delta a_{ii} \le a_{ii} + \gamma_{n+1}|\widehat{r}_i|^T|\widehat{r}_i|,$$

so that $\|\widehat{r}_i\|_2^2 \le (1-\gamma_{n+1})^{-1}a_{ii}$. Then, using the Cauchy–Schwarz inequality,

$$|\widehat{r}_i^T\widehat{r}_j| \le \|\widehat{r}_i\|_2\|\widehat{r}_j\|_2 \le (1-\gamma_{n+1})^{-1}(a_{ii}a_{jj})^{1/2},$$

giving

$$|\widehat{R}^T||\widehat{R}| \le (1-\gamma_{n+1})^{-1}dd^T \tag{10.8}$$

and the required bound for ΔA. □

Standard perturbation theory applied to Theorem 10.4 yields a bound of the form $\|x-\widehat{x}\|/\|x\| \le c_n u\kappa(A) + O(u^2)$. However, with the aid of Theorem 10.5 we can obtain a potentially much smaller bound. The idea is to write $A = DHD$ where $D = \mathrm{diag}(A)^{1/2}$, so that H has unit diagonal. van der Sluis's result (Corollary 7.6) shows that

$$\kappa_2(H) \le n \min_{F \text{ diagonal}} \kappa_2(FAF), \tag{10.9}$$

so D is nearly a condition-minimizing diagonal scaling. It follows that $\kappa_2(H) \le n\kappa_2(A)$ and that $\kappa_2(H) \ll \kappa_2(A)$ is possible if A is badly scaled. Note that $1 \le \|H\|_2 \le n$, since H is positive definite with $h_{ii} \equiv 1$.

Theorem 10.6 (Demmel, Wilkinson). *Let* $A = DHD \in \mathbb{R}^{n\times n}$ *be symmetric positive definite, where* $D = \mathrm{diag}(A)^{1/2}$*, and suppose Cholesky factorization successfully produces a computed solution* $\widehat{x}$ *to* $Ax = b$*. Then the scaled error* $D(x-\widehat{x})$ *satisfies*

$$\frac{\|D(x-\widehat{x})\|_2}{\|Dx\|_2} \le \frac{\kappa_2(H)\epsilon}{1-\kappa_2(H)\epsilon}, \tag{10.10}$$

where $\epsilon = 2n(1-\gamma_{n+1})^{-1}\gamma_{n+1}$.

Proof. Straightforward analysis shows that (cf. the proof of Theorem 9.4) $(A + \Delta A)\widehat{x} = b$, where

$$\Delta A = \Delta A_1 + \Delta_1\widehat{R} + \widehat{R}^T\Delta_2 + \Delta_1\Delta_2,$$

with $|DA_1| \le (1-\gamma_{n+1})^{-1}\gamma_{n+1}dd^T$ (by Theorem 10.5) and $|D_1| \le \mathrm{diag}(\gamma_i)|\widehat{R}^T|$, $|D_2| \le \mathrm{diag}(\gamma_{n-i+1})|\widehat{R}|$. Scaling with D, we have

$$(H + D^{-1}\Delta A D^{-1})D\widehat{x} = D^{-1}b,$$

and standard perturbation theory gives

$$\frac{\|D(x-\widehat{x})\|_2}{\|Dx\|_2} \le \frac{\kappa_2(H)\|D^{-1}\Delta A D^{-1}\|_2}{1-\kappa_2(H)\|D^{-1}\Delta A D^{-1}\|_2}.$$

But, using (10.8) and $\|D^{-1}dd^TD^{-1}\|_2 = \|ee^T\|_2 = n$, we have

$$\begin{aligned}\|D^{-1}\Delta A D^{-1}\|_2 &\le \frac{\gamma_{n+1}}{1-\gamma_{n+1}}\|D^{-1}dd^TD^{-1}\|_2 \\ &\quad + \mathrm{diag}(\gamma_i + \gamma_{n-i+1} + \gamma_i\gamma_{n-i+1})\|D^{-1}|\widehat{R}^T||\widehat{R}|D^{-1}\|_2 \\ &\le 2n(1-\gamma_{n+1})^{-1}\gamma_{n+1},\end{aligned}$$

using Lemma 3.3, which yields the result. $\square$

Care needs to be exercised when interpreting bounds that involve scaled quantities, but in this case the interpretation is relatively easy. Suppose that H is well conditioned and $\kappa_2(D)$ is large, which represents the artificial ill conditioning that the DHD scaling is designed to clarify. The vector $Dx = H^{-1}D^{-1}b$ is likely to have components that do not vary much in magnitude. Theorem 10.6 then guarantees that we obtain the components of Dx to good relative accuracy and this means that the components of x (which will vary greatly in magnitude) are obtained to good relative accuracy.

So far, our results have all contained the proviso that Cholesky factorization runs to completion—in other words, the results assume that the argument of the square root is always positive. Wilkinson [1092, 1968] showed that success is guaranteed if $20n^{3/2}\kappa_2(A)u \le 1$, that is, if A is not too ill conditioned. It would be nice to replace A in this condition by H, where $A = DHD$. Justification for doing so is that Algorithm 10.2 is scale invariant, in the sense that if we scale $A \leftarrow FAF$, where F is diagonal, then R scales to RF; moreover, if F comprises powers of the machine base, then even the rounding errors scale according to F. The following theorem gives an appropriate modification of Wilkinson's condition.

Theorem 10.7 (Demmel). *Let* $A = DHD \in \mathbb{R}^{n\times n}$ *be symmetric positive definite, when* $D = \mathrm{diag}(A)^{1/2}$. *If* $\lambda_{\min}(H) > n\gamma_{n+1}/(1-\gamma_{n+1})$ *then Cholesky factorization applied to* A *succeeds (barring underflow and overflow) and produces a nonsingular* $\widehat{R}$. *If* $\lambda_{\min}(H) \le -n\gamma_{n+1}/(1-\gamma_{n+1})$ *then the computation is certain to fail.*

Proof. The proof is by induction. Consider Algorithm 10.2. The first stage obviously succeeds and gives $\widehat{r}_{11} > 0$, since $a_{11} > 0$. Suppose the algorithm has successfully completed $k - 1$ stages, producing a nonsingular R_{k-1}, and consider equations (10.1)-(10.3) with n replaced by k. The kth stage can be completed, but may give a pure imaginary $\widehat{\beta}$ (it will if $fl(a - \widehat{r}^T\widehat{r}) < 0$). However, in the latter event, the error analysis of Theorem 10.5 is still valid! Thus we obtain $\widehat{R}_k$ satisfying $\widehat{R}_k^*\widehat{R}_k = A_k + DA_k$, $|DA_k| \le (1 - \gamma_{k+1})^{-1}\gamma_{k+1}d_k d_k^T$, where $d_k = [a_{11}^{1/2}, \dots, a_{kk}^{1/2}]^T$. Now, with $D_k = \mathrm{diag}(d_k)$, *we* have

$$\begin{aligned}\lambda_{\min}\big(D_k^{-1}(A_k + \Delta A_k)D_k^{-1}\big) &= \lambda_{\min}\big(H_k + D_k^{-1}\Delta A_k D_k^{-1}\big)\\ &\ge \lambda_{\min}(H_k) - \|D_k^{-1}\Delta A_k D_k^{-1}\|_2\\ &\ge \lambda_{\min}(H_k) - \frac{\gamma_{k+1}}{1-\gamma_{k+1}}\|ee^T\|_2\\ &\ge \lambda_{\min}(H) - k\frac{\gamma_{k+1}}{1-\gamma_{k+1}} > 0,\end{aligned}$$

using the interlacing of the eigenvalues [470, 1989, Cor. 8.1.4] and the condition of the theorem. Hence $D_k^{-1}(A_k + \Delta A_k)D_k^{-1}$ is positive definite, and therefore so is the congruent matrix $A_k + DA_k$, showing that $\widehat{R}_k$ must be real and nonsingular, as required to complete the induction.

If Cholesky succeeds, then, by Theorem 10.5, $D^{-1}(A + DA)D^{-1}$ is positive definite and so $0 < \lambda_{\min}(H) + \|D^{-1}\Delta A D^{-1}\|_2 \le \lambda_{\min}(H) + n(1-\gamma_{n+1})^{-1}\gamma_{n+1}$. Hence if $\lambda\min(H) < -n\gamma_{n+1}/(1-\gamma_{n+1})$ then the computation must fail. ◻

Note that, since $||H||2 \ge 1$, the condition for success of Cholesky factorization can be written as $\kappa_2(H)n\gamma_{n+1}/(1-\gamma_{n+1}) < 1$.

10.2. Sensitivity of the Cholesky Factorization

The Cholesky factorization has perturbation bounds that are similar to those for LU factorization, but of a simpler form thanks to the positive definiteness ($||A^{-1}||_2$ replaces $||U^{-1}||_2\,||L^{-1}||_2$ in the normwise bounds).

Theorem 10.8 (Sun). *Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite with the Cholesky factorization $A = R^TR$ and let AA be a symmetric matrix satisfying $||A^{-1}DA||_2 < 1$. Then $A + DA$ has the Cholesky factorization $A + DA = (R + DR)^T(R + DR)$, where*

$$\frac{\|\Delta R\|_F}{\|R\|_p} \le 2^{-1/2}\frac{\kappa_2(A)\epsilon}{1-\kappa_2(A)\epsilon}, \qquad \epsilon = \frac{\|\Delta A\|_F}{\|A\|_p}, \quad p = 2, F.$$

Moreover, if $\rho(|\widetilde{G}|) < 1$, *where* $\widetilde{G} = (R + DR)^{-T}DA(R + DR)^{-1}$, *then*

$$|DR| \leq \operatorname{triu}\big(|\widetilde{G}|(I - |\widetilde{G}|)^{-1}\big)|R + \Delta R|,$$

where triu(·) *denotes the upper triangular part.* □

Note that the Cholesky factor of $A_k = A(1{:}k,1{:}k)$ is R_k, and $\kappa_2(A_{k+1}) \geq \kappa_2(A_k)$ by the interlacing property of the eigenvalues. Hence if A_{k+1} (and hence A) is ill conditioned but A_k is well conditioned then R_k will be relatively insensitive to perturbations in A but the remaining columns of R will be much more sensitive.

10.3. Positive Semidefinite Matrices

If A is symmetric and positive semidefinite ($x^TAx \geq 0$ for all x) then a Cholesky factorization exists, but the theory and computation are more subtle than for positive definite A.

The questions of existence and uniqueness of a Cholesky factorization are answered by the following result.

Theorem 10.9. *Let* $A \in \mathbb{R}^{n\times n}$ *be positive semidefinite of rank* r. (a) *There exists at least one upper triangular* R *with nonnegative diagonal elements such that* $A = R^TR$. (b) *There is a permutation* P *such that* P^TAP *has a unique Cholesky factorization, which takes the form*

$$\Pi^TA\Pi = R^TR, \qquad R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix}, \tag{10.11}$$

where R_{11} *is* $r \times r$ *upper triangular with positive diagonal elements.*

Proof. (a): Let the symmetric positive semidefinite square root X of A have the QR factorization $X = QR$ with $r_{ii} \geq 0$. Then $A = X^2 = X^TX = R^TQ^TQR = R^TR$. (b): The algorithm with pivoting described below amounts to a constructive proof. □

Note that the factorization in part (a) is not in general unique. For example,

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \equiv \begin{bmatrix} 0 & 0 \\ \cos\theta & \sin\theta \end{bmatrix} \begin{bmatrix} 0 & \cos\theta \\ 0 & \sin\theta \end{bmatrix}.$$

For practical computations a factorization of the form (10.11) is needed, because this factorization so conveniently displays the rank deficiency. Such a factorization can be computed using an outer product Cholesky algorithm, comprising $r = \operatorname{rank}(A)$ stages. At the kth stage, a rank-1 matrix is subtracted from A so as to introduce zeros into positions $k{:}n$ in the kth row and column. Ignoring pivoting for the moment, at the start of the kth stage we have

$$A^{(k)} = \left(a_{ij}^{(k)}\right) = A - \sum_{i=1}^{k-1} r_i r_i^T = \begin{array}{c|cc} & k-1 & n-k+1 \\ k-1 & 0 & 0 \\ n-k+1 & 0 & A_k \end{array}, \tag{10.12}$$

where $r_i^T = [0, \ldots, 0, r_{ii}, \ldots, r_{in}]$. The reduction is carried one stage further by computing

$$\begin{aligned} r_{kk} &= \sqrt{a_{kk}^{(k)}}, \\ r_{kj} &= a_{kj}^{(k)}/r_{kk}, \qquad j = k+1\colon n, \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - r_{ki}r_{kj}, \quad i,j = k+1\colon n. \end{aligned}$$

Overall we have,

$$A = \sum_{i=1}^{r} r_i r_i^T = R^T R, \qquad R^T = [r_1, \ldots, r_r].$$

To avoid breakdown when $a_{kk}^{(k)}$ vanishes (or is negative because of rounding errors), pivoting is incorporated into the algorithm as follows. At the start of the kth stage an element $a_{ss}^{(k)} > 0$ ($s \geq k$) is selected as pivot, and rows and columns k and s of A_k, and the kth and sth elements of r_i, $i = 1\colon k-1$, are interchanged. The overall effect is to compute the decomposition (10.11), where the permutation P takes account of all the interchanges.

The standard form of pivoting is defined by

$$s = \min\left\{ j : a_{jj}^{(k)} = \max_{k \le i \le n} a_{ii}^{(k)} \right\}.$$

This is equivalent to complete pivoting in GE, since A_k is positive semidefinite so its largest element lies on the diagonal. We note for later reference that this pivoting strategy produces a matrix R that satisfies (cf. Problem 18.5)

$$r_{kk}^2 \geq \sum_{i=k}^{\min(j,r)} r_{ij}^2, \qquad j = k+1\colon n, \quad k = 1\colon r. \tag{10.13}$$

It will be convenient to denote by $\mathrm{cp}(A) := P^TAP$ the permuted matrix obtained from the Cholesky algorithm with complete pivoting.

10.3.1. Perturbation Theory

In this section we analyse, for a positive semidefinite matrix, the effect on the Cholesky factorization of perturbations in the matrix. This perturbation theory will be used in the error analysis of the next section.

Throughout this section A is assumed to be an $n \times n$ positive semidefinite matrix of rank r whose leading principal submatrix of order r is positive definite. For $k = 1{:}r$ we will write

$$A = \begin{array}{cc} & \begin{array}{cc} k & n-k \end{array} \\ \begin{array}{c} k \\ n-k \end{array} & \begin{bmatrix} A_{11} & A_{12} \\ A_{12}^T & A_{22} \end{bmatrix} \end{array} \tag{10.14}$$

and other matrices will be partitioned conformably.

We have the identity

$$A = \begin{array}{cc} & k \\ \begin{array}{c} k \\ n-k \end{array} & \begin{bmatrix} R_{11}^T \\ R_{12}^T \end{bmatrix} \end{array} [R_{11}, \quad R_{12}] + \begin{bmatrix} 0 & 0 \\ 0 & S_k(A) \end{bmatrix}, \tag{10.15}$$

where R_{11} is the Cholesky factor of A_{11}, $R_{12} = R_{11}^{-T} A_{12}$, and

$$S_k(A) = A_{22} - A_{12}^T A_{11}^{-1} A_{12}$$

is the Schur complement of A_{11} in A. Note that $S_r(A) \equiv 0$, so that for $k = r$, (10.15) is the (unique) Cholesky factorization of A. The following lemma shows how $S_k(A)$ changes when A is perturbed.

Lemma 10.10. *Let E be symmetric and assume $A_{11} + E_{11}$ is nonsingular. Then*

$$S_k(A+E) = S_k(A) + E_{22} - (E_{12}^T W + W^T E_{12}) + W^T E_{11} W + O(\|E\|^2), \tag{10.16}$$

where $W = A_{11}^{-1} A_{12}$.

Proof. We can expand

$$(A_{11} + E_{11})^{-1} = A_{11}^{-1} - A_{11}^{-1} E_{11} A_{11}^{-1} + A_{11}^{-1} E_{11} A_{11}^{-1} E_{11} A_{11}^{-1} + O(\|E_{11}\|^3).$$

The result is obtained by substituting this expansion into $S_k(A+E) = (A_{22} + E_{22}) - (A_{12} + E_{12})^T (A_{11} + E_{11})^{-1} (A_{12} + E_{12})$, and collecting terms. $\square$

Lemma 10.10 shows that the sensitivity of $S_k(A)$ to perturbations in A is governed by the matrix $W = A_{11}^{-1} A_{12}$. The question arises of whether, for a given A, the potential $\|W\|_2^2$ magnification of E indicated by (10.16) is attainable. For the no-pivoting strategy, $P = I$, the answer is trivially "yes", since we can take $E = \left[\begin{smallmatrix} \gamma I & 0 \\ 0 & 0 \end{smallmatrix}\right]$, with $|\gamma|$ small, to obtain $\|S_k(A+E) - S_k(A)\|_2 = \|W\|_2^2 \|E\|_2 + O(\|E\|_2^2)$. For complete pivoting, however, the answer is complicated by the possibility that the sequence of pivots will be different for $A+E$ than for A, in which case Lemma 10.10 is not applicable. Fortunately, a mild assumption on A is enough to rule out this technical difficulty, for small $\|E\|_2$. In the next lemma we redefine $A := \mathrm{cp}(A)$ in order to simplify the notation.

Lemma 10.11. *Let* $A := \mathrm{cp}(A)$. *Suppose that*

$$(S_i(A))_{11} > (S_i(A))_{jj}, \quad j = 2{:}\,n-i, \quad i = 0{:}\,r-1 \tag{10.17}$$

(*where* $S_0(A) := A$). *Then, for sufficiently small* $\|E\|_2$, $A+E = \mathrm{cp}(A+E)$. *For* $E = \left[\begin{smallmatrix} \gamma I & 0 \\ 0 & 0 \end{smallmatrix}\right]$, *with* $|\gamma|$ *sufficiently small,*

$$\|S_k(\mathrm{cp}(A+E)) - S_k(A)\|_2 = \|W\|_2^2\|E\|_2 + O(\|E\|_2^2).$$

Proof. Note that since $A = \mathrm{cp}(A)$, (10.17) simply states that there are no ties in the pivoting strategy (since $(S_i(A))_{11} \equiv a_{i+1,i+1}^{(i+1)}$ in (10.12)). Lemma 10.10 shows that $S_i(A+E) = S_i(A) + O(\|E\|_2)$, and so, in view of (10.17), for sufficiently small $\|E\|_2$ we have

$$\big(S_i(A+E)\big)_{11} > \big(S_i(A+E)\big)_{jj}, \qquad j = 2{:}\,n-i, \quad i = 0{:}\,r-1.$$

This shows that $A + E = \mathrm{cp}(A+E)$. The last part then follows from Lemma 10.10. □

We now examine the quantity $\|W\|_2 = \|A_{11}^{-1}A_{12}\|_2$. We show first that $\|W\|_2$ can be bounded in terms of the square root of the condition number of A_{11}.

Lemma 10.12. *If* A, *partitioned as in* (10.14), *is symmetric positive definite and* A_{11} *is positive definite then* $\|A_{11}^{-1}A_{12}\|_2 \le \sqrt{\|A_{11}^{-1}\|_2\|A_{22}\|_2}$.

Proof. Write $A_{11}^{-1}A_{12} = A_{11}^{-1/2}A_{11}^{-1/2}A_{12}$ and use $\|A_{11}^{-1/2}\|_2 = \|A_{11}^{-1}\|_2^{1/2}$, together with $\|A_{11}^{-1/2}A_{12}\|_2 = \|A_{12}^TA_{11}^{-1}A_{12}\|_2^{1/2} \le \|A_{22}\|_2^{1/2}$, which follows from the fact that the Schur complement $A_{22} - A_{12}^TA_{11}^{-1}A_{12}$ is positive semidefinite. □

Note that, by the arithmetic-geometric mean inequality $\sqrt{xy} \le (x+y)/2$ $(x, y \ge 0)$, we also have, from Lemma 10.12, $\|A_{11}^{-1}A_{12}\|_2 \le (\|A_{11}^{-1}\|_2 + \|A_{22}\|_2)/2$.

The inequality of Lemma 10.12 is attained for the positive semidefinite matrix

$$A = \begin{bmatrix} \alpha I_{k,k} & I_{k,n-k} \\ I_{n-k,k} & \alpha^{-1} I_{n-k,n-k} \end{bmatrix}, \qquad \alpha > 0,$$

where $I_{p,q}$ is the $p \times q$ identity matrix. This example shows that $\|W\|_2$ can be arbitrarily large. However, for $A := \mathrm{cp}(A)$, $\|W\|_2$ can be bounded solely in terms of n and k. The essence of the proof, in the next lemma, is that large elements in A_{11}^{-1} are countered by small elements in A_{12}. Hereafter we set $k = r$, the value of interest in the following sections.

Lemma 10.13. *Let A* := cp(*A*) *and set k* = *r. Then*

$$\|A_{11}^{-1}A_{12}\|_{2,F} \le \sqrt{\frac{1}{3}(n-r)(4^r-1)}. \tag{10.18}$$

There is a parametrized family of rank-r matrices A(q) = cp*(A(q)),* $\theta \in (0, \frac{\pi}{2}]$*, for which*

$$\|A_{11}(\theta)^{-1}A_{12}(\theta)\|_{2,F} \to \sqrt{\frac{1}{3}(n-r)(4^r-1)} \qquad \textit{as} \;\; \theta \to 0.$$

Proof. The proof is a straightforward computation. The matrix *A(O)* := *R(q)*^{T}*R(q),* where

$$R(\theta) = \mathrm{diag}(1, s, \ldots, s^{r-1}) \begin{bmatrix} 1 & -c & -c & \ldots & -c & -c & \ldots & -c \\ & 1 & -c & \ldots & -c & -c & \ldots & -c \\ & & 1 & & \vdots & \vdots & & \vdots \\ & & & \ddots & \vdots & \vdots & & \vdots \\ & & & & 1 & -c & \ldots & -c \end{bmatrix} \in \mathbb{R}^{r \times n}, \tag{10.19}$$

with *c* = cos*q*, *s* = sin *q*. This is the *r* × *n* version of Kahan's matrix (8.10). *R* satisfies the inequalities (10.13) (as equalities) and so *A(q)* = cp*(A(q)).* ☐

We conclude this section with a "worst-case" example for the Cholesky factorization with complete pivoting. Let *U*(*q*) = diag(*r*, *r*–1, ... 1)*R*(*q*), where *R(q)* is given by (10.19), and define the rank-*r* matrix *C(q)* = *U(q)*^{T}*U(q).* Then *C(q)* satisfies the conditions of Lemma 10.11. Also,

$$\begin{aligned} \|W\|_2 = \|C_{11}(\theta)^{-1}C_{12}(\theta)\|_2 &= \|U_{11}(\theta)^{-1}U_{12}(\theta)\|_2 = \|R_{11}(\theta)^{-1}R_{12}(\theta)\|_2 \\ &\to \sqrt{\frac{1}{3}(n-r)(4^r-1)} \qquad \text{as } \theta \to 0. \end{aligned}$$

Thus, from Lemma 10.11, for *E* = $\left[\begin{smallmatrix} \gamma I & 0 \\ 0 & 0 \end{smallmatrix}\right]$, with |γ| and *q* sufficiently small,

$$\|S_r\big(\mathrm{cp}\big(C(\theta) + E\big)\big)\|_2 \approx \frac{1}{3}(n-r)(4^r-1)\|E\|_2.$$

This example can be interpreted as saying that in exact arithmetic the residual after an *r*-stage Cholesky factorization of a semidefinite matrix *A* can overestimate the distance of *A* from the rank-r semidefinite matrices by a factor as large as (*n* – *r*)(4^r – 1)/3.

10.3.2. Error Analysis

In this section we present a backward error analysis for the Cholesky factorization of a positive semidefinite matrix. An important consideration is that a matrix of floating point numbers is very unlikely to be ‘(exactly” positive semidefinite; errors in forming or storing A will almost certainly render the smallest eigenvalue nonzero, and possibly negative. Therefore error analysis for a rank r positive semidefinite matrix may appear, at first sight, to be of limited applicability. One way around this difficulty is to state results for $A = A + DA$, where A is the matrix stored on the computer, A is positive semidefinite of rank r, and DA is a perturbation, which could represent the rounding errors in storing $\tilde{A}$, for example. However, if the perturbation AA is no larger than the backward error for the Cholesky factorization, then this extra formalism can be avoided by thinking of DA as being included in the backward error matrix. Hence for simplicity, we frame the error analysis for a positive semidefinite A.

The analysis makes no assumptions about the pivoting strategy, so A should be thought of as the pre-permuted matrix P^TAP.

Theorem 10.14. *Let A be an $n \times n$ symmetric positive semidefinite matrix of floating point numbers of rank $r \le n$. Assume that $A_{11} = A(1{:}r, 1{:}r)$ is positive definite with*

$$\lambda_{\min}(H_{11}) > r\gamma_{r+1}/(1-\gamma_{r+1}), \qquad (10.20)$$

where $A_{11} = D_{11}H_{11}D_{11}$ and $D_{11} = \mathrm{diag}(A_{11})^{1/2}$. Then, in floating point arithmetic, the Cholesky algorithm applied to A successfully completes r stages (barring underflow and overflow), and the computed $r \times n$ Cholesky factor $\widehat{R}_r$ satisfies

$$\|A - \widehat{R}_r^T\widehat{R}_r\|_2 \le 2r\gamma_{r+1}\|A\|_2(\|W\|_2+1)^2 + O(u^2), \qquad (10.21)$$

where $W = A_{11}^{-1}A_{12}$.

Proof. First, note that condition (10.20) guarantees successful completion of the first r stages of the algorithm by Theorem 10.7.

Analysis very similar to that leading to Theorem 10.3 shows that

$$A + E = \widehat{R}_r^T\widehat{R}_r + \widehat{A}^{(r+1)}, \qquad (10.22)$$

where

$$\widehat{R}_r = \begin{array}{c|cc} & r & n-r \\ r & \widehat{R}_{11} & \widehat{R}_{12} \end{array}, \qquad \widehat{A}^{(r+1)} = \begin{array}{c} \\ r \\ n-r \end{array}\!\!\begin{array}{c} \begin{array}{cc} r & n-r \end{array} \\ \begin{bmatrix} 0 & 0 \\ 0 & \widehat{S}_{r+1} \end{bmatrix} \end{array},$$

and

$$|E| \le \gamma_{r+1}(|\widehat{R}_r^T||\widehat{R}_r| + |\widehat{A}^{(r+1)}|). \qquad (10.23)$$

Taking norms in (10.23) and using the inequality $\| |B| \|_2 \le \sqrt{\operatorname{rank}(B)}\|B\|_2$, we obtain

$$\begin{aligned}\|E\|_2 &\le \gamma_{r+1}\big(r\|\widehat{R}_r^T\|_2\|\widehat{R}_r\|_2 + \sqrt{n-r}\|\widehat{A}^{(r+1)}\|_2\big)\\ &= \gamma_{r+1}\big(r\|\widehat{R}_r^T\widehat{R}_r\|_2 + \sqrt{n-r}\|\widehat{A}^{(r+1)}\|_2\big)\\ &= \gamma_{r+1}\big(r\|A+E-\widehat{A}^{(r+1)}\|_2 + \sqrt{n-r}\|\widehat{A}^{(r+1)}\|_2\big)\\ &\le \gamma_{r+1}\big(r\|A\|_2 + r\|E\|_2 + n\|\widehat{A}^{(r+1)}\|_2\big),\end{aligned}$$

which implies

$$\|E\|_2 \le \frac{\gamma_{r+1}}{1-r\gamma_{r+1}}\big(r\|A\|_2 + n\|\widehat{A}^{(r+1)}\|_2\big). \tag{10.24}$$

Our aim is to obtain an a priori bound for $\|A - \widehat{R}_r^T\widehat{R}_r\|_2$. It is clear from (10.22)-(10.24) that to do this we have only to bound $\|\widehat{A}^{(r+1)}\|_2$. To this end, we interpret (10.22) in such a way that the perturbation theory of §10.3.1 may be applied.

Equation (10.22) shows that $\widehat{S}_{r+1}$ is the true Schur complement for the matrix $A + E$ and that $A_{11} + E_{11} = \widehat{R}_{11}^T\widehat{R}_{11}$ is positive definite. Hence we can use Lemma 10.10 to deduce that

$$\begin{aligned}\|\widehat{A}^{(r+1)}\|_2 = \|\widehat{S}_{r+1}\|_2 &\le \|E_{22}\|_2 + 2\|E_{12}\|_2\|W\|_2 + \|W\|_2^2\|E_{11}\|_2 + O(u^2)\\ &\le \|E\|_2\big(\|W\|_2+1\big)^2 + O(u^2).\end{aligned}$$

Substituting from (10.24) we find that

$$\|\widehat{A}^{(r+1)}\|_2 \le r\gamma_{r+1}\|A\|_2(\|W\|_2+1)^2 + O(u^2).$$

Finally, using (10.22) and (10.24), we obtain

$$\|A - \widehat{R}_r^T\widehat{R}_r\|_2 \le 2r\gamma_{r+1}\|A\|_2(\|W\|_2+1)^2 + O(u^2). \qquad \square$$

Theorem 10.14 is just about the best result that could have been expected, because the bound (10.21) is essentially the same as the bound obtained on taking norms in Lemma 10.10. In other words, (10.21) simply reflects the inherent mathematical sensitivity of $A-R^TR$ to small perturbations in A.

We turn now to the issue of stability. Ideally, for A as defined in Theorem 10.14, the computed Cholesky factor R_r produced after r stages of the algorithm would satisfy

$$\|A - \widehat{R}_r^T\widehat{R}_r\|_2 \le c_n u\|A\|_2,$$

where c_n is a modest constant. Theorem 10.14 shows that stability depends on the size of $\|W\|_2 = \|A_{11}^{-1}A_{12}\|_2$ (to the extent that $\|W\|_2$ appears in a realistic bound for the backward error).

If no form of pivoting is used then $\|W\|_2$ can be arbitrarily large for fixed n (see §10.3.1) and the Cholesky algorithm must in this case be classed as unstable. But for complete pivoting we have from Lemma 10.13 the upper bound $\|W\|_2 \le (1/3(n-r)(4^r-1))^{1/2}$. Thus the Cholesky algorithm with complete pivoting is stable if r is small, but stability cannot be guaranteed, and seems unlikely in practice, if $\|W\|_2$ (and hence, necessarily, r and n) is large.

Numerical experiments show that $\|W\|_2$ is almost always small in practice (typically less than 10) [540, 1990]. However, it is easy to construct examples where $\|W\|_2$ is large. For example, if R is a Cholesky factor of A from complete pivoting then let $C = M(R)^T M(R)$, where $M(R)$ is the comparison matrix; C will usually have a much larger value of $\|W\|_2$ than A.

An important practical issue is when to terminate the Cholesky factorization of a semidefinite matrix. The LINPACK routine `xCHDC` proceeds with the factorization until a nonpositive pivot is encountered, that is, up to and including stage $k-1$, where k is the smallest integer for which

$$\widehat{a}_{ii}^{(k)} \le 0, \quad i = k\colon n. \tag{10.25}$$

Usually $k > r+1$, due to the effect of rounding errors.

A more sophisticated termination criterion is to stop as soon as

$$\|\widehat{S}_k\| \le \epsilon\|A\| \qquad \text{or} \qquad \widehat{a}_{ii}^{(k)} \le 0, \quad i = k\colon n, \tag{10.26}$$

for some readily computed norm and a suitable tolerance ϵ. This criterion terminates as soon as a stable factorization is achieved, avoiding unnecessary work in eliminating negligible elements in the computed Schur complement $\widehat{S}_k$. Note that $\|\widehat{S}_k\|$ is indeed a reliable order-of-magnitude estimate of the true residual, since $\widehat{S}_k$ is the only nonzero block of $\widehat{A}^{(k)}$ and, by (10.22) and (10.24), $A - \widehat{R}_{k-1}^T\widehat{R}_{k-1} = \widehat{A}^{(k)} - E$ with $\|E\| = O(u)(\|A\| + \|\widehat{A}^{(k)}\|)$.

Another possible stopping criterion is

$$\max_{k\le i\le n} \widehat{a}_{ii}^{(k)} \le \epsilon\widehat{a}_{11}^{(1)}. \tag{10.27}$$

This is related to (10.26) in that if A (pre-permuted) and $\widehat{A}_k$ are positive semidefinite then $a_{11}^{(1)} = \max_{i,j} |a_{ij}| \approx \|A\|_2$, and similarly $\max_{k\le i\le n} \widehat{a}_{ii}^{(k)} \approx \|\widehat{S}_k\|_2$. Note that (10.27) bounds $\kappa_2(\widehat{R}_{k-1})$, since if (10.27) holds first at the kth stage then, using Theorem 8.13,

$$\frac{\kappa_2(\widehat{R}_{k-1})}{(k-1)^{1/2}\,2^{k-2}} \le \frac{|\widehat{r}_{11}|}{|\widehat{r}_{k-1,k-1}|} = \left(\frac{\widehat{a}_{11}^{(1)}}{\widehat{a}_{k-1,k-1}^{(k-1)}}\right)^{1/2} \le \epsilon^{-1/2}.$$

Practical experience shows that the criteria (10.26) and (10.27) with $\epsilon = nu$ both work well, and much more effectively than (10.25) [540, 1990]. We favour (10.27) because of its negligible cost.

10.4. Symmetric Indefinite Matrices and Diagonal Pivoting Method

Let $A \in \mathbb{R}^{n\times n}$ be symmetric but indefinite, that is, $(x^TAx)(y^TAy) < 0$ for some x and y. How can we solve $Ax = b$ efficiently?

Gaussian elimination with partial pivoting (GEPP) can be used to compute the factorization $PA = LU$, but it does not take advantage of the symmetry to reduce the cost and storage. We might try to construct a factorization $A = LDL^T$, where L is unit lower triangular and D is diagonal. But this factorization may not exist, even if symmetric pivoting is allowed, and if it does exist its computation may be unstable. For example, consider

$$A \equiv PAP^T = \begin{bmatrix} \epsilon & 1 \\ 1 & \epsilon \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/\epsilon & 1 \end{bmatrix} \begin{bmatrix} \epsilon & 0 \\ 0 & \epsilon - 1/\epsilon \end{bmatrix} \begin{bmatrix} 1 & 1/\epsilon \\ 0 & 1 \end{bmatrix}.$$

There is arbitrarily large element growth for $0 < \epsilon \ll 1$, and the factorization does not exist for $\epsilon = 0$.

The most popular approach for solving symmetric indefinite systems is to use a block LDL^T factorization

$$PAP^T = LDL^T,$$

where L is unit lower triangular and D is block diagonal with 1×1 or 2×2 diagonal blocks. This factorization is essentially a symmetric block form of GE, with pivoting. Note that by Sylvester's inertia theorem, A and D have the same inertia[13], which is easily determined from D (*see* Problem 10.11).

To begin the computation of the factorization we choose a permutation P and an integer $s = 1$ or 2 so that

$$\Pi A \Pi^T = \begin{array}{c} s \\ n-s \end{array} \begin{bmatrix} E & C^T \\ C & B \end{bmatrix},$$

with E nonsingular. Then we compute the factorization

$$\Pi A \Pi^T = \begin{bmatrix} I_s & 0 \\ CE^{-1} & I_{n-s} \end{bmatrix} \begin{bmatrix} E & 0 \\ 0 & B - CE^{-1}C^T \end{bmatrix} \begin{bmatrix} I_s & E^{-1}C^T \\ 0 & I_{n-s} \end{bmatrix}.$$

---

[13] The inertia of a symmetric matrix is an ordered triple $\{i_+, i_-, i_0\}$, where i_+ is the number of positive eigenvalues, i_- the number of negative eigenvalues, and i_0 the number of zero eigenvalues.

This process is repeated on the $(n - s) \times (n - s)$ Schur complement

$$\tilde{A} = B - CE^{-1}C^T.$$

The cost of the method is $n^3/3$ flops (the same as the cost of Cholesky factorization of a positive definite matrix) plus the cost of determining the permutations 17. This method for computing the block LDL^T factorization is called the *diagonal pivoting method.* It can be thought of as a generalization of Lagrange's method for reducing a quadratic form to diagonal form (devised by Lagrange in 1759 and rediscovered by Gauss in 1823) [763, 1961, p. 371].

One conceivable difficulty with the diagonal pivoting method can be disposed of immediately. If a nonsingular pivot matrix E of dimension 1 or 2 cannot be found, then all 1 × 1 and 2 × 2 principal submatrices of the symmetric matrix A are singular, and this is easily seen to imply that A is the zero matrix.

The strategy for choosing 17 is crucial for achieving stability. A suitable modification of the error analysis for block LU factorization (Theorem 12.4) tells us that, provided linear systems involving 2 x 2 pivots are solved in a normwise backward-stable way, the condition $\|L\| \, \|D\| \, \|L^T\| \le c_n\|A\|$, for a modest constant c_n, is sufficient to ensure stability. A key requirement, therefore, is to choose the pivot E so that the Schur complement $\tilde{A}$ is suitably bounded, since D is made up of elements of Schur complements. We describe two suitable pivoting strategies.

10.4.1. Complete Pivoting

Bunch and Parlett [166, 1971] devised the following strategy for choosing 17. It suffices to describe the interchanges for the first stage of the factorization.

```
Let $\mu_0 = \max_{i,j} |a_{ij}|$, $\mu_1 = \max_i |a_{ii}|$, and choose $a \in (0, 1)$.
if $\mu_1 \ge a\mu_0$
    Set $s = 1$, and choose $P$ so that $|e_{11}| = \mu_1$.
else
    Set $s = 2$, and choose $P$ so that $|e_{21}| = \mu_0$.
end
```

Note that μ_1 is the best 1 × 1 pivot under symmetric permutations and μ_0 is the pivot that would be chosen by GE with complete pivoting. This strategy says "as long as there is a diagonal pivot element not much smaller than the complete pivot, choose it as a 1 × 1 pivot", that is, "choose a 1 × 1 pivot whenever possible". If the strategy dictates the use of a 2 × 2 pivot then that pivot E is indefinite (see Problem 10.11).

It remains to determine a. This is done by minimizing a bound on the element growth. For the following analysis we assume that the interchanges have already been done. If $s = 1$ then

$$\widetilde{a}_{ij} = b_{ij} - c_{i1}\frac{1}{e_{11}}c_{1j} \quad \Rightarrow \quad |\widetilde{a}_{ij}| \le \mu_0 + \frac{\mu_0^2}{\mu_1} \le \Big(1 + \frac{1}{\alpha}\Big)\mu_0.$$

Now consider the case $s = 2$. The (i, j) element of the Schur complement $\tilde{A} = B - CE^{-1}C^T$ is

$$\widetilde{a}_{ij} = b_{ij} - \begin{bmatrix} c_{i1} & c_{i2} \end{bmatrix} E^{-1} \begin{bmatrix} c_{j1} \\ c_{j2} \end{bmatrix}. \tag{10.28}$$

Now

$$E^{-1} = \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix}^{-1} = \frac{1}{\det(E)} \begin{bmatrix} e_{22} & -e_{12} \\ -e_{21} & e_{11} \end{bmatrix}$$

and, using the symmetry of E,

$$\det(E) = e_{11}e_{22} - e_{21}^2 = e_{11}e_{22} - \mu_0^2 \le \mu_1^2 - \mu_0^2 \le (\alpha^2 - 1)\mu_0^2.$$

Since $\alpha \in (0, 1)$, we have $|\det(E)| \ge (1 - \alpha^2)\mu_0^2$. Thus

$$|E^{-1}| \le \frac{1}{(1-\alpha^2)\mu_0} \begin{bmatrix} \alpha & 1 \\ 1 & \alpha \end{bmatrix}.$$

Since $|c_{ij}| \le \mu_0$, we obtain from (10.28)

$$|\widetilde{a}_{ij}| \le \mu_0 + \frac{2(1+\alpha)\mu_0^2}{(1-\alpha^2)\mu_0} = \left(1 + \frac{2}{1-\alpha}\right)\mu_0.$$

To determine α we equate the maximum growth for two $s = 1$ steps with that for one $s = 2$ step:

$$\left(1 + \frac{1}{\alpha}\right)^2 = 1 + \frac{2}{1-\alpha},$$

which reduces to the quadratic equation $4\alpha^2 - \alpha - 1 = 0$. We require the positive root

$$\alpha = \frac{1+\sqrt{17}}{8} \approx 0.64.$$

The analysis guarantees a growth factor bound of $(1+\alpha^{-1})^{n-1} = (2.57)^{n-1}$. This bound is pessimistic, however; a much more detailed analysis by Bunch [158, 1971] shows that the growth factor is no more than $3.07(n-1)^{0.446}$ times larger than the bound (9.13) for LU factorization with complete pivoting—a very satisfactory result. Strictly speaking, bounding the growth factor bounds only $\|D\|$, not $\|L\|$. But it is easy to show that for $s = 1$ and 2 no element of CE^{-1} exceeds $\max\{1/\alpha, 1/(1-\alpha)\}$ in absolute value, and so $\|L\|$ is bounded independently of A.

Since complete pivoting requires the whole active submatrix to be searched at each stage, it needs up to $n^3/6$ comparisons, and so the method is rather expensive.

10.4.2. Partial Pivoting

Bunch and Kaufman [164, 1977] devised a pivoting strategy for the diagonal pivoting method that requires only $O(n^2)$ comparisons. At each stage it searches at most two columns and so is analogous to partial pivoting for LU factorization. The strategy contains several logical tests. As before, we describe the pivoting for the first stage only. Recall that s denotes the size of the pivot block.

```
        Choose $a \in (0, 1)$.
        $\lambda := \|A(2{:}n, 1)\|_\infty$
        If $\lambda = 0$ there is nothing to do on this stage of the elimination.
        $r := \min\{i \ge 2: |a_{i1}| = \lambda\}$
        if $|a_{11}| \ge \lambda$
(1)         $s = 1$, $P = I$
        else
            $\sigma := \left\| \begin{bmatrix} A(1{:}r-1, r) \\ A(r+1{:}n, r) \end{bmatrix} \right\|_\infty$
            if $|a_{11}|s \ge a\lambda^2$
(2)             $s = 1$, $P = I$
            else if $|a_{rr}| > as$
(3)             $s = 1$ and choose $P$ to swap rows and columns 1 and $r$.
            else
(4)             $s = 2$ and choose $P$ to swap rows and columns 2 and $r$,
                so that $|(PAP^T)_{21}| =: \lambda$.
            end
        end
```

To understand the algorithm it helps to consider the matrix

$$\begin{bmatrix} a_{11} & \dots & \lambda & \dots & \dots & \dots \\ \vdots & & \vdots & & & \\ \lambda & \dots & a_{rr} & \dots & \sigma & \dots \\ \vdots & & \vdots & & & \\ \vdots & & \sigma & & & \\ \vdots & & \vdots & & & \end{bmatrix},$$

and to note that the pivot is one of a_{11}, a_{rr}, and $\begin{bmatrix} a_{11} & \lambda \\ \lambda & a_{rr} \end{bmatrix}$ (or, rather, since $\lambda := |a_{r1}|$, this matrix with λ replaced by a_{r1}).

To bound the element growth reconsider each case in turn, noting that for cases (1) and (2) the elements of the Schur complement are given by[14]

$$\widetilde{a}_{ij} = a_{ij} - \frac{a_{i1}a_{1j}}{a_{11}}.$$

Case (1):

$$|\widetilde{a}_{ij}| \le |a_{ij}| + \frac{1}{\alpha}|a_{1j}| \le \left(1 + \frac{1}{\alpha}\right) \max_{i,j} |a_{ij}|.$$

Case (2): Using symmetry,

$$|\widetilde{a}_{ij}| \le |a_{ij}| + \frac{\lambda^2}{|a_{11}|} \le |a_{ij}| + \frac{\sigma}{\alpha} \le \left(1 + \frac{1}{\alpha}\right) \max_{i,j} |a_{ij}|.$$

Case (3): The original a_{rr} is now the pivot, and $|a_{rr}| \ge a s$, so

$$|\widetilde{a}_{ij}| \le |a_{ij}| + \frac{\sigma}{|a_{rr}|}|a_{1j}| \le \left(1 + \frac{1}{\alpha}\right) \max_{i,j} |a_{ij}|.$$

Case (4): This is where we use a 2 x 2 pivot, which, after the interchanges, is $E = (P^T AP)(1{:}2,1{:}2) = \left[\begin{smallmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{smallmatrix}\right]$. Now

$$|\det(E)| = |a_{r1}^2 - a_{11}a_{rr}| \ge \lambda^2 - |a_{11}|\alpha\sigma \ge \lambda^2 - \alpha(\alpha\lambda^2) = \lambda^2(1-\alpha^2).$$

The elements of the Schur complement $\tilde{A} = B - CE^{-1}C^T$ are given by

$$\widetilde{a}_{ij} = b_{ij} - \det(E)^{-1} \begin{bmatrix} a_{i1} & a_{ir} \end{bmatrix} \begin{bmatrix} a_{rr} & -a_{r1} \\ -a_{r1} & a_{11} \end{bmatrix} \begin{bmatrix} a_{j1} \\ a_{jr} \end{bmatrix},$$

so

$$\begin{aligned} |\widetilde{a}_{ij}| &\le |b_{ij}| + (\lambda^2(1-\alpha^2))^{-1} \begin{bmatrix} \lambda & \sigma \end{bmatrix} \begin{bmatrix} \alpha\sigma & \lambda \\ \lambda & |a_{11}| \end{bmatrix} \begin{bmatrix} \lambda \\ \sigma \end{bmatrix} \\ &\le |b_{ij}| + \frac{2\lambda^2\sigma(1+\alpha)}{\lambda^2(1-\alpha^2)} = |b_{ij}| + \frac{2\sigma}{1-\alpha} \le \left(1 + \frac{2}{1-\alpha}\right) \max_{i,j} |a_{ij}|. \end{aligned}$$

This analysis shows that the bounds for the element growth for $s = 1$ and $s = 2$ are the same as the respective bounds for the complete pivoting strategy. Hence, using the same reasoning, we again choose $a = (1+\sqrt{17})/8$.

The growth factor for the partial pivoting strategy is bounded by $(2.57)^{n-1}$. As for GEPP, large growth factors do not seem to occur in practice. But unlike for GEPP, no example is known for which the bound is attained [164, 1977]; see Problem 10.18.

---

[14] We commit a minor abuse of notation, in that in the rest of this section $\tilde{a}_{ij}$ should really be $\tilde{a}_{i-1,j-1}$ $(s = 1)$ or $\tilde{a}_{i-2,j-2}$ $(s = 2)$.

As noted in the previous subsection, a bound on the growth factor ρ_n does not in itself ensure stability. Indeed although $\|D\|/\|A\|$ is bounded for partial pivoting, $\|L\|/\|A\|$ can be arbitrarily large for fixed n; see Problem 10.15. Higham [559, 1995] gives a detailed error analysis of the diagonal pivoting method with an arbitrary pivoting strategy, under the assumption that computed solutions to linear systems involving 2 × 2 pivots have a small componentwise relative backward error. The conclusions are that the computed factors satisfy

$$\widehat{L}\widehat{D}\widehat{L}^T = P(A+\Delta A)P^T, \quad |\Delta A| \le p_1(n)u\big(|A| + P^T|\widehat{L}||\widehat{D}||\widehat{L}^T|P\big) + O(u^2)$$

and the computed solution to a linear system $Ax = b$ satisfies

$$(A+\Delta A)\widehat{x} = b, \quad |\Delta A| \le p_2(n)u\big(|A| + P^T|\widehat{L}||\widehat{D}||\widehat{L}^T|P\big) + O(u^2),$$

where p_1 and p_2 are linear polynomials. For the partial pivoting strategy, Higham shows that if linear systems involving 2 × 2 pivots are solved by GEPP or by use of the explicit inverse, then the computed solutions do indeed have a small componentwise relative backward error, and that, moreover,

$$\|\,|L||D||L^T|\,\|_M \le 36n\rho_n\|A\|_M,$$

where $\|A\|_M = \max_{i,j} |a_{ij}|$. Thus the diagonal pivoting method with partial pivoting is stable if the growth factor is small.

10.5. Nonsymmetric Positive Definite Matrices

The notion of positive definiteness can be extended to nonsymmetric matrices. A nonsymmetric matrix $A \in \mathbb{R}^{n\times n}$ is *positive definite* if $x^TAx > 0$ for all $x \ne 0$. This is equivalent to the condition that the symmetric part A_S of A is positive definite, where $A \equiv A_S + A_K$ with $A_S = (A + A^T)/2$ and $A_K = (A - A^T)/2$. A positive definite matrix clearly has nonsingular leading principal submatrices, and so has an LU factorization, $A = LU$. It can even be shown that pivots u_{ii} are positive. However, there is no guarantee that the factorization is stable without pivoting, as the example $\left[\begin{smallmatrix}\epsilon & 1\\ -1 & \epsilon\end{smallmatrix}\right]$ shows. The standard error analysis for LU factorization applies (Theorems 9.3 and 9.4), and so the question is whether $|\widehat{L}||\widehat{U}|$ can be suitably bounded. Golub and Van Loan [469, 1979] show that, for the exact LU factors,

$$\|\,|L||U|\,\|_F \le n\|A_S + A_K^TA_S^{-1}A_K\|_2. \tag{10.29}$$

Let $\chi(A) = \|A_S + A_K^TA_S^{-1}A_K\|_2\|A_S^{-1}\|_2$, which is just $\kappa_2(A)$ when A is symmetric. Mathias [731, 1992] shows that $\|\,|L||U|\,\|_F$ (involving now the computed LU factors) is at most a factor $1 + 30un^{3/2}\chi(A)$ times larger than the upper bound in (10.29), and that the LU factorization (without pivoting) succeeds if $24n^{3/2}\chi(A)u \leq 1$.

These results show that it is safe not to pivot provided that the symmetric part of A is not too ill conditioned relative to the norm of the skew-symmetric part. If A is symmetric ($A_K = 0$) then we recover the results for symmetric positive definite matrices.

10.6. Notes and References

André-Louis Cholesky (1875-1918) was a French military officer involved in geodesy and surveying in Crete and North Africa. In some books his name is misspelled "Choleski". Details of Cholesky's life-and a discussion about the pronunciation of his name!-can be found in the electronic mail magazine NA-Digest, volume 90, 1990, issues 7, 8, 10–12, and 24; see, in particular, the biography [22, 1922]. Cholesky's work was published posthumously on his behalf by Benoit [91, 1924].

The properties of the Cholesky factorization are intimately associated with the properties of the Schur complement, as is apparent from some of the proofs in this chapter. The same is true for GE in general. An excellent survey of the Schur complement, containing historical comments, theory, and applications, is given by Cottle [248, 1974].

For results on the Cholesky factorization in Hilbert space see Power [841, 1986].

A book by George and Liu [438, 1981] is devoted to the many practical issues in the implementation of Cholesky factorization for the solution of sparse symmetric positive definite systems.

There is no floating point error analysis of Cholesky factorization in Wilkinson's books, but he gives a detailed analysis in [1092, 1968], showing that $\widehat{R}^T\widehat{R} = A+E$, with $\|E\|_2 \leq 2.5n^{3/2}u\|A\|_2$. It is unfortunate that this paper is in a rather inaccessible proceedings, because it is a model of how to phrase and interpret an error analysis. Meinguet [747, 1983] and Sun [973, 1992] give componentwise backward error bounds similar to those in Theorems 10.3 and 10.4. Kielbasinski [657, 1987] reworks Wilkinson's analysis to improve the constant.

The fact that $\kappa_2(H)$ can replace the potentially much larger $\kappa_2(A)$ in the forward error bound for the Cholesky method was stated informally and without proof by Wilkinson [1092, 1968, p. 638]. Demmel [283, 1989] made this observation precise and explored its implications; Theorems 10.5, 10.6, and 10.7 are taken from [283, 1989].

The bounds in Theorem 10.8 are from Sun [971, 1991], [972, 1992]. Similar bounds are given by Stewart [944, 1977], [951, 1993], Barrlund [71, 1991], and Sun [973, 1992]. A perturbation bound that can be much smaller than the normwise one in Theorem 10.8 is derived and explored by Chang and Paige [198, 1995]. Perturbation results of a different flavour, including one for structured perturbations of the form of *DA* in Theorem 10.5, are given by Drmac, Omladc, and Veselic [321, 1994].

The perturbation and error analysis of §10.3 for semidefinite matrices is from Higham [540, 1990], where in a perturbation result for the *QR* factorization with column pivoting is also given. For an application in optimization that makes use of Cholesky factorization with complete pivoting and the analysis of §10.3.1 see Forsgren, Gill, and Murray [384, 1995].

Fletcher and Powell [382, 1974] describe several algorithms for updating an LDL^T factorization of a symmetric positive definite *A* when *A* is modified by a rank-1 matrix. They give detailed componentwise error analysis for some of the methods.

An excellent way to test whether a given symmetric matrix *A* is positive (semi) definite is to attempt to compute a Cholesky factorization. This test is less expensive than computing the eigenvalues and is numerically stable. Indeed, if the answer "yes" is obtained, it is the right answer for a nearby matrix, whereas if the answer is "no" then *A must* be close to an indefinite matrix. See Higham [535, 1988] for an application of this definiteness test. An algorithm for testing the definiteness of a Toeplitz matrix is developed by Cybenko and Van Loan [260, 1986], as part of a more complicated algorithm. According to Kerr [654, 1990], misconceptions of what is a sufficient condition for a matrix to be positive (semi) definite are rife in the engineering literature (for example, that it suffices to check the definiteness of all 2 × 2 submatrices). See also Problem 10.8. For some results on definiteness tests for Toeplitz matrices, see Makhoul [722, 1991].

A major source of symmetric indefinite linear systems is the least squares problem, because the augmented system is symmetric indefinite; see Chapter 19. Other sources of such systems are interior methods for solving constrained optimization problems (see Forsgren, Gill, and Shinnerl [385, 1996], Turner [1030, 1991], and Wright [1115, 1992]) and linearly constrained optimization problems (see Gill, Murray, Saunders, and Wright [445, 1990], [446, 1991]).

The idea of using a block LDL^T factorization with some form of pivoting for symmetric indefinite matrices was first suggested by Kahan in 1965 [166, 1971]. Bunch and Parlett [166, 1971] developed the complete pivoting strategy and Bunch [158, 1971] proved its stability. Bunch [160, 1974] discusses a rather expensive partial pivoting strategy that requires repeated scalings. Bunch and Kaufman [164, 1977] found the efficient partial pivoting strategy presented here, which is the one now widely used, and Bunch, Kaufman and Parlett [165, 1976] give an Algol code implementing the diagonal pivoting method with this pivoting strategy. Dongarra, Duff, Sorensen, and van der Vorst [315, 1991, §5.4.5] show how to develop a partitioned version of the diagonal pivoting method with partial pivoting.

Liu [709, 1987] shows how to incorporate a threshold into the Bunch–Kaufinan partial pivoting strategy for sparse symmetric matrices; see also Duff et al. [326, 1991]. The partial pivoting strategy and variants of it described by Bunch and Kaufman [164, 1977] do not preserve band structure, but the fill-in depends on the number of 2 × 2 pivots, which is bounded by the number of negative eigenvalues (see Problem 10.11). Jones and Patrick [615, 1993], [616, 1994] show how to exploit this fact.

The complete and partial pivoting strategies of Bunch et al. use a fixed number of tests to determine each pivot. Another possibility is to prescribe growth bounds corresponding to 1 × 1 and 2 × 2 pivots and to search in some particular order for a pivot satisfying the bound. Fletcher [375, 1976] uses this approach to define a pivoting strategy that usually requires only $O(n^2)$ operations. Duff, Reid, and co-workers apply the same approach to the diagonal pivoting method for sparse matrices, where sparsity considerations also influence the choice of pivot [331, 1979], [326, 1991]; their Fortran codes MA27 [329, 1982] and MA47 [330, 1995] implement the methods.

Gill, Murray, Ponceleón, and Saunders [443, 1992] show how for sparse, symmetric indefinite systems the diagonal pivoting factorization can be used to construct a (positive definite) preconditioned for an iterative method.

Another method for solving symmetric indefinite systems is Aasen's method [1, 1971], which employs the factorization $PAP^T = LTL^T$, where L is unit lower triangular and T is tridiagonal. It is competitive with the diagonal pivoting method in terms of speed. Barwell and George [77, 1976] compare the performance of Fortran codes for several methods for solving symmetric indefinite systems, including the diagonal pivoting method and Aasen's method.

Dax and Kaniel [270, 1977] propose computing a factorization $PAP^T = LDL^T$ for symmetric indefinite matrices by an extended form of Gaussian elimination in which extra row operations are used to "build up" a pivot element prior to the elimination operations; here, L is unit lower triangular and D is diagonal. A complete pivoting strategy for determining the permutation P is described in [270, 1977] and partial pivoting strategies in Dax [268, 1982].

Analogues of the factorization for symmetric matrices exist for skew-symmetric matrices; see Bunch [161, 1982].

Bunch [159, 1971] shows how to scale a symmetric matrix so that in every nonzero row and column the largest magnitude of an element is 1.

10.6.1. LAPACK

Driver routines `xPOSV` (simple) and `xPOSVX` (expert) use the Cholesky factorization to solve a symmetric (or Hermitian) positive definite system of linear equations with multiple right-hand sides. (There are corresponding routines for packed storage, in which one triangle of the matrix is stored in a one-dimensional array: PP replaces PO in the names.) The expert driver incorporates iterative refinement, condition estimation, and backward and forward error estimation and has an option to scale the system $AX = B$ to $(D^{-1}AD^{-1})DX = D^{-1}B$, where $D = \mathrm{diag}(a_{ii}^{1/2})$. Modulo the rounding errors in computing and applying the scaling, the scaling has no effect on the accuracy of the solution prior to iterative refinement, in view of Theorem 10.6. The Cholesky factorization is computed by the routine xPOTRF, which uses a partitioned algorithm that computes R a block row at a time. The drivers xPTSV and xPTSVX for symmetric positive definite tridiagonal matrices use LDL^T factorization. LAPACK does not currently contain a routine for Cholesky factorization of a positive semidefinite matrix, but there is such a routine in LINPACK (xCHDC).

Driver routines xSYSV (simple) and xSYSVX (expert) use the block LDL^T factorization (computed by the diagonal pivoting method) with partial pivoting to solve a symmetric indefinite system of linear equations with multiple right-hand sides. For Hermitian matrices the corresponding routines are xHESV (simple) and xHESVX (expert). (Variants of these routines for packed storage have names in which SP replaces SY and HP replaces HE.) The expert drivers incorporate iterative refinement, condition estimation, and backward and forward error estimation. The factorization is computed by the routine xSYTRF or xHETRF.

Problems

10.1. Show that if $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite then

$$|a_{ij}| < \sqrt{a_{ii}a_{jj}} \quad \text{for all } i \neq j.$$

What does this statement imply about $\max_{i,j} |a_{ij}|$?

10.2. If A is a symmetric positive definite matrix, how would you compute $x^TA^{-1}x$?

10.3. Let $y = \left(c - \sum_{i=1}^{k-1} a_ib_i\right)^{1/2}$ be evaluated in floating point arithmetic in any order. Show that

$$\widehat{y}^2(1+\theta_{k+1}) = c - \sum_{i=1}^{k-1} a_ib_i(1+\theta_{k-1}^{(i)}),$$

where $|\theta_{k-1}^{(i)}| \le \gamma_{k-1}$ for all i, and $|q_{k+1}| \le \gamma_{k+1}$.

10.4. Let $A \in \mathbb{R}^{n\times n}$ be symmetric positive definite. Show that the reduced submatrix B of order n—1 at the end of the first stage of GE is also symmetric positive definite. Deduce that $0 < a_{kk}^{(k)} \le a_{kk}^{(k-1)} \le \cdots \le a_{kk}^{(1)} = a_{kk}$ and hence that the growth factor $\rho_n = 1$.

10.5. Show that the backward error result (10.6) for the solution of a symmetric positive definite linear system by Cholesky factorization implies

$$(A + \Delta A)\widehat{x} = b, \qquad \|\Delta A\|_M \le 2\gamma_{n+1}(1 - \gamma_{n+1})^{-1}\|A\|_M,$$

where $\|A\|_M = \max_{i,j} |a_{ij}|$ (which is not a consistent matrix norm—see §6.2). The significance of this result is that the bound for $\|DA\|_M/\|A\|_M$ contains a linear polynomial in n, rather than the quadratic that appears for the 2-norm in (10.7).

10.6. Let $A = \mathrm{cp}(A) \in \mathbb{R}^{n\times n}$ be positive semidefinite of rank r and suppose it has the Cholesky factorization (10.11) with $P = I$. Show that $Z = [W, -I]^T$ is a basis for the null space of A, where $W = A_{11}^{-1}A_{12}$.

10.7. Prove that (10.13) holds for the Cholesky decomposition with complete pivoting.

10.8. Give an example of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ for which the leading principal submatrices A_k satisfy $\det(A_k) \ge 0$, $k = 1{:}n$, but A is not positive semidefinite (recall that $\det(A_k) > 0$, $k = 1{:}n$, implies that A is positive definite). State a condition on the minors of A that is both necessary and sufficient for positive semidefiniteness.

10.9. Suppose the outer product Cholesky factorization algorithm terminates at the (k+1)st stage (see (10.15)), with a negative pivot in the $(k + 1, k + 1)$ position. Show how to construct a direction of negative curvature for A (a vector p such that $p^TAp < 0$).

10.10. What is wrong with the following argument? A positive semidefinite matrix is the limit of a positive definite one as the smallest eigenvalue tends to zero. Theorem 10.3 shows that Cholesky factorization is stable for a positive definite matrix, and therefore, by continuity, it must be stable for a positive semidefinite matrix, implying that Theorem 10.14 is unnecessarily weak (since $\|W\|_2$ can be large).

10.11. Consider the diagonal pivoting method applied to a symmetric matrix. Show that with complete pivoting or partial pivoting any 2 × 2 pivot is indefinite. Hence give a formula for the inertia in terms of the block sizes of the block diagonal factor. Show how to avoid overflow in computing the inverse of a 2 × 2 pivot.

10.12. Describe the effect of applying the diagonal pivoting method with partial pivoting to a 2 × 2 symmetric matrix.

10.13. What factorization is computed if the diagonal pivoting method with partial pivoting is applied to a symmetric positive definite matrix?

10.14. (Sorensen and Van Loan; see [315, 1991, §5.3.2]) Suppose the partial pivoting strategy for the diagonal pivoting method is modified by redefining

$$\sigma = \|A(:,r)\|_\infty$$

(thus "$s_{\mathrm{new}} = \max(\sigma_{\mathrm{old}}, |a_{rr}|)$"). Show that the same growth factor bound holds as before and that for a positive definite matrix no interchanges are done and only 1 × 1 pivots are used.

10.15. Let

$$A = \begin{bmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix},$$

where $0 < \epsilon < 1$, and suppose the diagonal pivoting method is applied to A, yielding a factorization $PAP^T = LDL^T$. Show that with partial pivoting $\|L\|_\infty$ is unbounded as $\epsilon \to 0$, whereas with complete pivoting $\|L\|_\infty$ is bounded independently of ϵ.

10.16. Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

be nonsymmetric positive definite. Show that the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is also positive definite. In other words, show that GE preserves positive definiteness.

10.17. A matrix of the form

$$A = \begin{bmatrix} H & B^T \\ B & -G \end{bmatrix},$$

where $H \in \mathbb{R}^{n\times n}$ and $G \in \mathbb{R}^{m\times m}$ are symmetric positive definite has been called a *symmetric quasidefinite matrix* by Vanderbei [1047, 1995]. Show that (a) A is nonsingular, (b) for any permutation P, P^TAP has an LU factorization, (c) AS is nonsymmetric positive definite, where $S = \mathrm{diag}(I, -I)$. (This last property reduces the question of the stability of an LDL^T factorization of A to that of the stability of the LU factorization of a nonsymmetric positive definite matrix, for which see §10.5. This reduction has been pointed out and exploited by Gill, Saunders, and Shinnerl [448, 1996].)

10.18. (RESEARCH PROBLEM) Is the growth factor bound $(2.57)^{n-1}$ for the diagonal pivoting method with partial pivoting attainable? If not, how big can the growth factor be? Similarly, what is a sharp bound for the complete pivoting growth factor?

10.19. (RESEARCH PROBLEM) Bound the growth factor for Aasen's method [1, 1971].

Chapter 11

Iterative Refinement

The ILLIAC's memory is sufficient to accommodate a system of 39 equations when used with Routine 51. The additional length of Routine 100 *restricts to 37 the number of equations that it can handle. With* 37 *equations the operation time of Routine* 100 *is about 4 minutes per iteration.*

—JAMES N. SNYDER, *On the improvement of the Solutions to a Set of Simultaneous Linear Equations Using the ILLIAC (1955)*

In a short mantissa computing environment the presence of an iterative improvement routine can significantly widen the class of solvable $Ax = b$ problems.

— GENE H. GOLUB and CHARLES F. VAN LOAN, *Matrix Computations (1989)*

Most problems involve inexact input data and obtaining a highly accurate solution to an imprecise problem may not be justified.

— J. J. DONGARRA, J. R. BUNCH, C. B. MOLER, and G. W. STEWART, *LINPACK Users' Guide (1979)*

Iterative refinement is an established technique for improving a computed solution $\widehat{x}$ to a linear system $Ax = b$. The process consists of three steps:

1. Compute $r = b - A\widehat{x}$.
2. Solve $Ad = r$.
3. Update $y = \widehat{x} + d$.

(Repeat from step 1 if necessary, with $\widehat{x}$ replaced by y).

If there were no rounding errors in the computation of r, d, and y, then y would be the exact solution to the system. The idea behind iterative refinement is that if r and d are computed accurately enough then some improvement in the accuracy of the solution will be obtained. The economics of iterative refinement are favorable for solvers based on a factorization of A, because the factorization used to compute $\widehat{x}$ can be reused in the second step of the refinement.

Traditionally, iterative refinement is used with Gaussian elimination (GE), and r is computed in extended precision before being rounded to working precision. Iterative refinement for GE was used in the 1940s on desk calculators, but the first thorough analysis of the method was given by Wilkinson in 1963 [1088, 1963]. The behaviour of iterative refinement for GE is usually summarized as follows: if double precision is used in the computation of r, and A is not too ill conditioned, then the iteration produces a solution correct to working precision and the rate of convergence depends on the condition number of A. In the next section we give a componentwise analysis of iterative refinement that confirms this summary and provides some further insight.

11.1. Convergence of Iterative Refinement

Let $A \in \mathbb{R}^{n\times n}$ be nonsingular and let $\widehat{x}$ be a computed solution to $Ax = b$. Define $x_1 = \widehat{x}$ and consider the following iterative refinement process: $r_i = b - Ax_i$ (precision $\overline{u}$), solve $Ad_i = r_i$ (precision u), $x_{i+1} = x_i + d_i$ (precision u), $i = 1, 2, \ldots$. For traditional iterative refinement, $\overline{u} := u^2$. Note that in this chapter subscripts specify members of a vector sequence, not vector elements.

We henceforth define r_i, d_i, and x_i to be the *computed* quantities (to avoid a profusion of hats). The only assumption we will make on the solver is that the computed solution $\widehat{y}$ to a system $Ay = c$ satisfies

$$(A + \Delta A)\widehat{y} = c, \qquad \|A^{-1}\Delta A\|_\infty < 1. \tag{11.1}$$

Thus the solver need not be LU factorization or even a factorization method.

The page or so of analysis that follows is straightforward but tedious. The reader is invited to jump straight to (11.4), at least on first reading.

Consider first the computation of r_i. There are two stages. First, $s_i = fl(b - Ax_i) = b - Ax_i + \Delta s_i$ is formed in the (possibly) extended precision $\overline{u}$, so that $|\Delta s_i| \le \overline{\gamma}_{n+1}(|b| + |A||x_i|)$ (cf. (3.10)), where $\overline{\gamma}_k \le k\overline{u}/(1 - k\overline{u})$. Second, the residual is rounded to the working precision: $r_i = fl(s_i) = s_i + f_i$, where $|f_i| \le u|s_i|$. Hence

$$r_i = b - Ax_i + \Delta r_i, \qquad |\Delta r_i| \le u|b - Ax_i| + (1+u)\overline{\gamma}_{n+1}(|b| + |A||x_i|).$$

By writing $x_i = x + (x_i - x)$, we obtain the bound

$$|\Delta r_i| \le [u + (1+u)\overline{\gamma}_{n+1}]|A||x - x_i| + 2(1+u)\overline{\gamma}_{n+1}|A||x|. \tag{11.2}$$

For the second step we have, by (11.1), $(A + \Delta A_i)d_i = r_i$. Now write

$$(A + \Delta A_i)^{-1} = \bigl(A(I + A^{-1}\Delta A_i)\bigr)^{-1} =: (I + F_i)A^{-1},$$

where, since $\theta_i := \| A^{-1}\Delta A_i \|_\infty < 1$, $\|F_i\|_\infty \le \theta_i/(1 - \theta_i)$. Hence

$$d_i = (I + F_i)A^{-1}r_i = (I + F_i)(x - x_i + A^{-1}\Delta r_i). \tag{11.3}$$

For the last step,

$$\begin{aligned} x_{i+1} &= x_i + d_i + \Delta x_i, \\ |\Delta x_i| &\le u|x_i + d_i| \le u(|x - x_i| + |x| + |d_i|). \end{aligned}$$

Using (11.3) we have

$$x_{i+1} - x = F_i(x - x_i) + (I + F_i)A^{-1}\Delta r_i + \Delta x_i.$$

Hence

$$\begin{aligned} |x_{i+1} - x| &\le |F_i||x - x_i| + (I + |F_i|)|A^{-1}||\Delta r_i| + u|x - x_i| + u|x| + u|d_i| \\ &\le |F_i||x - x_i| + (I + |F_i|)|A^{-1}||\Delta r_i| + u|x - x_i| + u|x| \\ &\qquad + u(I + |F_i|)(|x - x_i| + |A^{-1}||\Delta r_i|) \\ &= \bigl((1+u)|F_i| + 2uI\bigr)|x - x_i| + (1+u)(I + |F_i|)|A^{-1}||\Delta r_i| + u|x|. \end{aligned}$$

Substituting the bound for $|\Delta r_i|$ from (11.2) gives

$$\begin{aligned} |x_{i+1} - x| \le\ & \bigl((1+u)|F_i| + 2uI\bigr)|x - x_i| \\ & + (1+u)\bigl(u + (1+u)\overline{\gamma}_{n+1}\bigr)(I + |F_i|)|A^{-1}||A||x - x_i| \\ & + 2(1+u)^2\overline{\gamma}_{n+1}(I + |F_i|)|A^{-1}||A||x| + u|x| \\ =:\ & G_i|x - x_i| + g_i. \end{aligned} \tag{11.4}$$

Note that

$$\begin{aligned} G_i &\approx |F_i| + u(I + |F_i|)|A^{-1}||A|, \\ g_i &\approx 2\overline{\gamma}_{n+1}(I + |F_i|)|A^{-1}||A||x| + u|x|. \end{aligned}$$

As long as A is not too ill conditioned and the solver is not too unstable, we have $\|G_i\|_\infty < 1$, which means that the error contracts until we reach a point at which the g_i term becomes significant. The limiting normwise accuracy, that is, the minimum size of $\|x - x_i\|_\infty/\|x\|_\infty$, is roughly $\|g_i\|_\infty/\|x\|_\infty \approx 2n\overline{u}\,\mathrm{cond}(A,x) + u$. Moreover, if $2n\overline{u}(I + |F_i|)|A^{-1}||A||x| \le \mu u|x|$ for some m, then we can expect to obtain a componentwise relative error of order mu, that is, $\min_i$ $|x - x_i| \lesssim \mu u|x|$.

We concentrate now on the case where the solver uses LU factorization. In the traditional use of iterative refinement, $\overline{u} = u^2$, and one way to summarize our findings is as follows.

Theorem 11.1 (mixed precision iterative refinement). *Let iterative refinement be applied to the nonsingular linear system* $Ax = b$, *using LU factorization and with residuals computed in double the working precision. Let* $h = u\|\,|A^{-1}||L||U|\,\|_\infty$, *where* L *and* U *are the computed LU factors of* A. *Then, provided* h *is sufficiently less than* 1, *iterative refinement reduces the error by a factor approximately* h *at each stage, until* $\|x - \widehat{x}_i\|_\infty/\|x\|_\infty \approx u$.

This theorem is stronger than the standard results in the literature, which have $\kappa_\infty(A)u$ in place of h. We can have $h \ll \kappa_\infty(A)u$, since h is independent of the row scaling of A (modulo changes in the pivot sequence). For example, if $|\widehat{L}||\widehat{U}| \approx |A|$ then $h \approx \mathrm{cond}(A)u$, and $\mathrm{cond}(A)$ can be arbitrarily smaller than $\kappa_\infty(A)$.

Consider now the case where $\overline{u} = u$, which is called *fixed precision iterative refinement.* We have an analogue of Theorem 11.1.

Theorem 11.2 (fixed precision iterative refinement). *Let iterative refinement in fixed precision be applied to the nonsingular linear system* $Ax = b$ *of order* n, *using LU factorization. Let* $h = u\|\,|A^{-1}||L||U|\,\|_\infty$, *where* L *and* $\widehat{U}$ *are the computed LU factors of* A. *Then, provided* h *is sufficiently less than* 1, *iterative refinement reduces the error by a factor approximately* h *at each stage, until* $\|x - \widehat{x}_i\|_\infty/\|x\|_\infty \lesssim 2n\,\mathrm{cond}(A, x)u$.

The key difference between mixed and fixed precision iterative refinement is that in the latter case a relative error of order u is no longer ensured. But we do have a relative error bound of order $\mathrm{cond}(A, x)u$. This is a stronger bound than holds for the original computed solution $\widehat{x}$, for which we can say only that

$$\frac{\|x - \widehat{x}\|_\infty}{\|x\|_\infty} \lesssim 2nu\frac{\|\,|A^{-1}||\widehat{L}||\widehat{U}||x|\,\|_\infty}{\|x\|_\infty}$$

(this bound is obtained by applying Theorems 7.4 and 9.4, or from (11.4) with $i = 0$!). In fact, a relative error bound of order $\mathrm{cond}(A, x)u$ is the best we can possibly expect if we do not use higher precision, because it corresponds to the uncertainty introduced by making componentwise relative perturbations to A of size u (again, see Theorem 7.4); this level of uncertainty is usually present, because of errors in computing A or in rounding its elements to floating point form.

The gist of this discussion is that iterative refinement is beneficial even if residuals are computed only at the working precision. This fact became widely appreciated only after the publication of Skeel's 1980 paper [920, 1980]. One reason for the delayed appreciation may be that comments such as that made by Forsythe and Moler, "It is absolutely essential that the residuals r_k be computed with a higher precision than that of the rest of the computation" [396, 1967, p. 49], were incorrectly read to mean that without the use of higher precision no advantage *at all* could be obtained from iterative refinement. In the next section we will see that fixed precision iterative refinement does more than just produce a $\mathrm{cond}(A, x)u$-bounded forward error for LU factorization—it brings componentwise backward stability as well.

11.2. Iterative Refinement Implies Stability

We saw in the last section that fixed precision iterative refinement can improve the accuracy of a solution computed by GE. The question arises of what the refinement process does to the backward error. To answer this question we give a general backward error analysis that is applicable to a wide class of linear equation solvers. Throughout this section, "iterative refinement" means fixed precision iterative refinement.

We assume that the computed solution $\widehat{x}$ to $Ax = b$ satisfies

$$|b - A\widehat{x}| \le u\big(g(A,b)|\widehat{x}| + h(A,b)\big), \tag{11.5}$$

where $g : \mathbb{R}^{n\times(n+1)} \to \mathbb{R}^{n\times n}$ and $h : \mathbb{R}^{n\times(n+1)} \to \mathbb{R}^{n}$ have nonnegative entries. The functions g and h may depend on n and u as well as on the data A and b. We also assume that the residual $r = b - A\widehat{x}$ is computed in such a way that

$$|\widehat{r} - r| \le ut(A,b,\widehat{x}), \tag{11.6}$$

where $t : \mathbb{R}^{n\times(n+2)} \to \mathbb{R}^{n}$ is nonnegative. If r is computed in the conventional way, then we can take

$$t(A,b,\widehat{x}) = \frac{\gamma_{n+1}}{u}(|A||\widehat{x}| + |b|). \tag{11.7}$$

First we give an asymptotic result that does not make any further assumptions on the linear equation solver.

Theorem 11.3. *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. Suppose the linear system $Ax = b$ is solved in floating point arithmetic using a solver S together with one* *step of iterative refinement. Assume that the computed solution* $\widehat{x}$ *produced by S satisfies (11.5) and that the computed residual? satisfies (11.6). Then the corrected solution* $\widehat{y}$ *satisfies*

$$|b - A\widehat{y}| \le u\big(h(A,\widehat{r}) + t(A,b,\widehat{y}) + |A||\widehat{y}|\big) + uq, \tag{11.8}$$

where $q = O(u)$ *if* $t(A,b,\widehat{x}) - t(A,b,\widehat{y}) = O(\|\widehat{x} - \widehat{y}\|_\infty)$.

Proof. The residual $r = b - A\widehat{x}$ of the original computed solution $\widehat{x}$ satisfies

$$|r| \le u\big(g(A,b)|\widehat{x}| + h(A,b)\big). \tag{11.9}$$

The computed residual is $\widehat{r} = r + Dr$, where $|Dr| \le ut(A,b,\widehat{x})$. The computed correction d satisfies

$$A\widehat{d} = \widehat{r} + f_1, \qquad |f_1| \le u\big(g(A,\widehat{r})|\widehat{d}| + h(A,\widehat{r})\big). \tag{11.10}$$

Finally, for the corrected solution we have

$$\widehat{y} = fl(\widehat{x} + \widehat{d}) = \widehat{x} + \widehat{d} + f_2, \qquad |f_2| \le u(|\widehat{x}| + |\widehat{d}|). \tag{11.11}$$

Collecting together the above results we obtain

$$b - A\widehat{y} = b - A\widehat{x} - A\widehat{d} - Af_2 = \widehat{r} - \Delta r - A\widehat{d} - Af_2 = -f_1 - \Delta r - Af_2.$$

Hence

$$\begin{aligned} |b - A\widehat{y}| &\le u\big(g(A,\widehat{r})|\widehat{d}| + h(A,\widehat{r})\big) + ut(A,b,\widehat{x}) + u|A|(|\widehat{x}| + |\widehat{d}|) \qquad (11.12)\\ &= u\big(h(A,\widehat{r}) + t(A,b,\widehat{y}) + |A||\widehat{y}|\big) + uq, \end{aligned}$$

where

$$q = t(A,b,\widehat{x}) - t(A,b,\widehat{y}) + g(A,\widehat{r})|\widehat{d}| + |A|(|\widehat{x}| - |\widehat{y}| + |\widehat{d}|).$$

The claim about the order of q follows since $\widehat{x} - \widehat{y}$, $|\widehat{x}| - |\widehat{y}|$, and $\widehat{d}$ are all of order u. □

Theorem 11.3 shows that, to first order, the componentwise relative backward error $w_{|A|,|b|}$ will be small after one step of iterative refinement as long as $h(A,\widehat{r})$ and $t(A,b,\widehat{y})$ are bounded by a modest scalar multiple of $|A||\widehat{y}| + |b|$. This is true for t if the residual is computed in the conventional way (see (11.7)), and in some cases we may take $h \equiv 0$, as shown below. Note that the function g of (11.5) does not appear in the first-order term of (11.8). This is the essential reason why iterative refinement improves stability: potential instability manifested in g is suppressed by the refinement stage.

A weakness of Theorem 11.3 is that the bound (11.8) is asymptotic. Since a strict bound for q is not given, it is difficult to draw firm conclusions about the size of $w_{|A|,|b|}$. The next result overcomes this drawback, at the cost of some specialization (and a rather long proof).

We introduce a measure of ill scaling of the vector $|B||x|$,

$$\sigma(B,x) = \frac{\max_i(|B||x|)_i}{\min_i(|B||x|)_i}.$$

Theorem 11.4. *Under the conditions of Theorem 11.3, suppose that $g(A, b) = G|A|$ and $h(A, b) = H|b|$, where $G, H \in \mathbb{R}^{n\times n}$ have nonnegative entries, and that the residual is computed in the conventional manner. Then there is a function*

$$f(t_1,t_2) \approx \big(t_2(t_1+n+1)/\operatorname{cond}(A^{-1}) + 2(t_1+n+2)^2(1+ut_2)^2\big)/(n+1)$$

such that if

$$\operatorname{cond}(A^{-1})\sigma(A,\widehat{y}) \le \big(f(\|G\|_\infty,\|H\|_\infty)u\big)^{-1}$$

then

$$|b - A\widehat{y}| \le 2\gamma_{n+1}|A||\widehat{y}|.$$

Proof. As with the analysis in the previous section, this proof can be skipped without any real loss of understanding. From (11.12) in the proof of Theorem 11.3, using the formula (11.7) for t, we have

$$|b - A\widehat{y}| \le uH|\widehat{r}| + \gamma_{n+1}|b| + (\gamma_{n+1}+u)|A||\widehat{x}| + u(I+G)|A||\widehat{d}|. \tag{11.13}$$

The inequality (11.9) implies

$$|b| - |A||\widehat{x}| \le |b - A\widehat{x}| \le u\big(G|A||\widehat{x}| + H|b|\big),$$

or $(I - uH)|b| \le (I + uG)|A||\widehat{x}|$. If $u\|H\|_\infty < 1/2$ (say) then $I - uH$ is nonsingular with a nonnegative inverse satisfying $\|(I - uH)^{-1}\|_\infty \le 2$ and we can solve for $|b|$ to obtain $|b| \le (I - uH)^{-1}(I + uG)|A||\widehat{x}|$. It follows from this relation and consideration of the rest of the proof that the simplifying step of replacing b by 0 in the analysis has little effect on the bounds—it merely produces unimportant perturbations in f in the statement of the theorem. Making this replacement in (11.13) and approximating $\gamma_{n+1} + u \approx \gamma_{n+1}$, we have

$$|b - A\widehat{y}| \le uH|\widehat{r}| + \gamma_{n+1}|A||\widehat{x}| + u(I+G)|A||\widehat{d}|. \tag{11.14}$$

Our task is now to bound $|A||\widehat{x}|$, $|\widehat{r}|$, and $|A||\widehat{d}|$ in terms of $|\widehat{y}|$. By manipulating (11.11) we obtain the inequality

$$|\widehat{x}| \le (1-u)^{-1}\big(|\widehat{y}| + (1+u)|\widehat{d}|\big) \approx |\widehat{y}| + |\widehat{d}|. \tag{11.15}$$

Also, we can bound $|\widehat{r}|$ by

$$|\widehat{r}| \le |r| + |\Delta r| \le u\bigl(G|A||\widehat{x}| + H|b|\bigr) + \gamma_{n+1}\bigl(|A||\widehat{x}| + |b|\bigr),$$

and dropping the $|b|$ terms and using (11.15) gives

$$|\widehat{r}| \le (uG + \gamma_{n+1}I)|A||\widehat{x}| \le (uG + \gamma_{n+1}I)|A|(|\widehat{y}| + |\widehat{d}|). \tag{11.16}$$

Substituting from (11.15) and (11.16) into (11.14) we find

$$\begin{aligned} |b - A\widehat{y}| \le\ & \bigl(\gamma_{n+1}I + uH(uG + \gamma_{n+1}I)\bigr)|A||\widehat{y}| \\ & + \bigl(\gamma_{n+1}I + u(I + G) + uH(uG + \gamma_{n+1}I)\bigr)|A||\widehat{d}| \\ =:\ & (\gamma_{n+1}I + M_1)|A||\widehat{y}| + M_2|A||\widehat{d}|, \end{aligned} \tag{11.17}$$

where

$$\begin{aligned} \|M_1\|_\infty &\le u\|H\|_\infty(u\|G\|_\infty + \gamma_{n+1}), \\ \|M_2\|_\infty &\le \gamma_{n+2} + u\|G\|_\infty + u\|H\|_\infty(u\|G\|_\infty + \gamma_{n+1}). \end{aligned}$$

Now from (11.10), making use of (11.16),

$$\begin{aligned} |\widehat{d}| &\le |A^{-1}|(|\widehat{r}| + uG|A||\widehat{d}| + uH|\widehat{r}|) \\ &\le |A^{-1}|\bigl((I + uH)(uG + \gamma_{n+1}I)|A|(|\widehat{y}| + |\widehat{d}|) + uG|A||\widehat{d}|\bigr). \end{aligned}$$

After premultiplying by $|A|$ this may be rearranged as

$$(I - uM_3)|A||\widehat{d}| \le u|A||A^{-1}|\, M_4\, |A||\widehat{y}|, \tag{11.18}$$

where

$$\begin{aligned} M_3 &= |A||A^{-1}|\bigl((I + uH)(G + (\gamma_{n+1}/u)I) + G\bigr), \\ M_4 &= (I + uH)(G + (\gamma_{n+1}/u)I). \end{aligned}$$

Using $\gamma_{n+1}/u \le (n+1)/(1-(n+1)u) \approx n+1$, we have the bounds

$$\begin{aligned} \|M_3\|_\infty &\le \operatorname{cond}(A^{-1})(\|G\|_\infty + n + 1)(2 + u\|H\|_\infty), \\ \|M_4\|_\infty &\le (\|G\|_\infty + n + 1)(1 + u\|H\|_\infty). \end{aligned}$$

If $u\|M_3\|_\infty < 1/2$ (say) then $(I - uM3)^{-1} \ge 0$ with $\|(I - uM3)\text{-}1\|_\infty \le 2$ and we can rewrite (11.18) as

$$|A||\widehat{d}| \le u(I - uM_3)^{-1}|A||A^{-1}|\, M_4\, |A||\widehat{y}|. \tag{11.19}$$

Substituting this bound into (11. 17) we obtain

$$\begin{aligned}|b - A\widehat{y}| &\le \left(\gamma_{n+1}I + M_1 + uM_2(I - uM_3)^{-1}|A||A^{-1}|M_4\right)|A||\widehat{y}|\\ &=: (\gamma_{n+1}I + M_5)|A||\widehat{y}|\\ &\le \omega|A||\widehat{y}|,\end{aligned}$$

where

$$\omega = \gamma_{n+1} + \|M_5\|_\infty \sigma(A, \widehat{y})$$

(see Problem 11.1). Finally, we bound $\|M_5\|_\infty$ Writing, $g = \|G\|_\infty$, $h = \|H\|_\infty$, we have

$$\begin{aligned}\|M_5\|_\infty \le u^2gh + uh\gamma_{n+1} + 2u(\gamma_{n+2} + ug + u^2gh + uh\gamma_{n+1})\\ \times \operatorname{cond}(A^{-1})(g + n + 1)(1 + uh)\end{aligned}$$

and this expression is approximately bounded by $u^2(h(g + n + 1) + 2(g + n + 2)^2(1 + uh)^2 \operatorname{cond}(A^{-1}))$. Requiring $\|M_5\|_\infty\sigma(A, \widehat{y})$ not to exceed γ_{n+1} leads to the result. $\square$

Theorem 11.4 says that as long as A is not too ill conditioned, $|A||\widehat{y}|$ is not too badly scaled ($\operatorname{cond}(A^{-1})\sigma(A, \widehat{y})$ is not too large), and the solver is not too unstable ($f(\|G\|_\infty, \|H\|_\infty)$ is not too large), then $w_{|A|,|b|} \le 2\gamma_{n+1}$ after one step of iterative refinement. Note that the term γ_{n+1} $|A||\widehat{y}|$ in (11.20) comes from the error bound for evaluation of the residual, so this bound for w is about the smallest we could expect to prove.

Let us apply Theorem 11.4 to GE with or without pivoting. If there is pivoting, assume (without loss of generality) that no interchanges are required. Theorem 9.4 shows that we can take

$$g(A, b) \approx 2n|\widehat{L}||\widehat{U}|, \qquad h(A, b) = 0,$$

where $\widehat{L}$, $\widehat{U}$ are the computed LU factors of A. To apply Theorem 11.4 we use $A \approx LU$ and write

$$g(A, b) \approx 2n|\widehat{L}||\widehat{L}^{-1}A| \le 2n|\widehat{L}||\widehat{L}^{-1}||A|,$$

which shows that we can take

$$G = 2n|\widehat{L}||\widehat{L}^{-1}|, \qquad f(\|G\|_\infty, \|H\|_\infty) \approx 8n\|\,|\widehat{L}||\widehat{L}^{-1}|\,\|_\infty^2.$$

Without pivoting the growth factor-type term $\|\,|\widehat{L}||\widehat{L}^{-1}|\,\|_\infty$ is unbounded, but with partial pivoting it cannot exceed 2^n and is typically $O(n)$ [1019, 1990] .

We can conclude that, for GE with partial pivoting (GEPP), one step of iterative refinement will usually be enough to yield a small componentwise

Table 11.1. $w_{|A|,|b|}$ values for A = orthog(25).

$\mathrm{cond}(A^{-1})\sigma(A,x) = 3.02\mathrm{e}1$
$\mathrm{cond}(A) = 2.09\mathrm{e}1, \quad \kappa_\infty(A) = 2.10\mathrm{e}1$

GEPP	GE	QR
2.53e-16	4.61e-7	4.54e-16
4.59e-17	1.56e-13	5.31e-17
	4.34e-17	

relative backward error as long as A is not too ill conditioned and $|A||\widehat{y}|$ is not too badly scaled. Without pivoting the same holds true with the added proviso that the computation of the original $\widehat{x}$ must not be too unstable.

These results for GE are very similar to those of Skeel [920, 1980]. The main differences are that Skeel's analysis covers an arbitrary number of refinement steps with residuals computed in single or double precision, his analysis is specific to GE, and his results involve $s(A, x)$ rather than $s(A, \widehat{y})$.

One interesting problem remains: to reconcile Theorem 11.4 with Theorem 11.2. Under the conditions of Theorem 11.4 the componentwise relative backward error is small after one step of iterative refinement, so the forward error is certainly bounded by a multiple of $\mathrm{cond}(A, x)u$. How can this be shown (for GE) using the analysis of §11.1? An explanation is nontrivial-see Problem 11.2.

We will see applications of Theorems 11.3 and 11.4 to other types of linear equation solver in Chapters 18, 19, and 21.

Tables 11.1–11.3 show the performance of fixed precision iterative refinement for GE without pivoting, GEPP, and Householder QR factorization (see §18.6). The matrices are from the Test Matrix Toolbox (see Appendix E), and may be summarized as follows. Clement(n) is tridiagonal with zero diagonal entries; orthog(n) is a symmetric and orthogonal matrix, and gfpp(n) is a matrix for which the growth factor for GEPP is maximal. In each case the right-hand side b was chosen as a random vector from the uniform distribution on [0, 1]. We report the componentwise relative backward errors for the initial solution and the refined iterates (refinement was terminated when $\omega_{|A|,|b|}(\widehat{y}) \le u$). GEPP performs as predicted by both our and Skeel's analyses. In fact, iterative refinement converges in one step even when $q(A, x) := \mathrm{cond}(A^{-1})s(A, x)$ exceeds u^{-1} in the examples reported and in most others we have tried. GE also achieves a small componentwise relative backward error, but can require more than one refinement step, even when $q(A, x)$ is small.

Table 11.2. $w_{|A|,|b|}$ values for A = clement (50).

$$\mathrm{cond}(A^{-1})\sigma(A,x) = 2.40\mathrm{e}18$$
$$\mathrm{cond}(A) = 1.44\mathrm{e}6, \quad \kappa_\infty(A) = 3.50\mathrm{e}7$$

GEPP	GE	QR
3.88e-15	Fail	1.43e-7
7.74e-17		1.04e-15
		6.71e-17

Table 11.3. $w_{|A|,|b|}$ *values for* A = gfpp(50).

$$\mathrm{cond}(A^{-1})\sigma(A,x) = 4.51\mathrm{e}2$$
$$\mathrm{cond}(A) = 50, \quad \kappa_\infty(A) = 50$$

GEPP	GE	QR
8.03e-4	8.03e-4	3.22e-16
8.06e-17	8.06e-17	3.82e-17

11.3. Notes and References

Wilkinson [1088, 1963] gave a detailed analysis of iterative refinement in a kind of scaled fixed point arithmetic called block-floating arithmetic. Moler [765, 1967] extended the analysis to floating point arithmetic. Very readable analyses of iterative refinement are given in the books by Forsythe and Moler [396, 1967, §22] and Stewart [941, 1973, §4.5].

As we mentioned in §9.10, as early as 1948 Wilkinson had written a program for the ACE to do GEPP and iterative refinement. Other early implementations of iterative refinement are in a code for the University of Illinois' ILLIAC by Snyder [932, 1955], the Algol code of McKeeman [745, 1962], and the Algol codes in the Handbook [138, 1966], [729, 1966]. Some of the machines for which these codes were intended could accumulate inner products in extended precision, and so were well suited to mixed precision iterative refinement.

Interest in fixed precision iterative refinement was sparked by two papers that appeared in the late 1970s. Jankowski and Wozniakowski [610, 1977] proved that an arbitrary linear equation solver is made normwise backward stable by the use of fixed precision iterative refinement, as long as the solver is not too unstable to begin with and A is not too ill conditioned. Skeel [920, 1980] analysed iterative refinement for GEPP and showed that one step of refinement yields a small componentwise relative backward error, as long as $\mathrm{cond}(A^{-1})s(A, x)$ is not too large.

The analysis in §11.1 extends existing results in the literature. The analysis in §11.2 is from Higham [549, 1991].

The quantity $s(A, x)$ appearing in Theorem 11.4 can be interpreted as follows. Consider a linear system $Ax = b$ for which $(|A||x|)_i = 0$ for some i. While the componentwise relative backward error $w_{|A|,|b|}^{(x)}$ of the exact solution x is zero, an arbitrarily small change to a component x_j where $a_{ij} \neq 0$ yields $w_{|A|,|b|}(x + Dx) \geq 1$. Therefore solving $Ax = b$ to achieve a small componentwise relative backward error can be regarded as an ill-posed problem when $|A||x|$ has a zero component. The quantity $s(A, x)$ reflects this ill-posedness because it is large when $|A||x|$ has a relatively small component.

For a lucid survey of both fixed and mixed precision iterative refinement and their applications, see Björck [111, 1990]. For particular applications of fixed precision iterative refinement, see Govaerts and Pryce [475, 1990] and Jankowski and Wozniakowski [611, 1985].

By increasing the precision from one refinement iteration to the next it is possible to compute solutions to arbitrarily high accuracy, an idea first suggested by Stewart in an exercise [941, 1973, pp. 206–207]. For algorithms, see Kielbasinski [656, 1981] and Smoktunowicz and Sokolnicka [931, 1984].

There are a number of practical issues to attend to when implementing iterative refinement. Mixed precision iterative refinement cannot be implemented in a portable way when the working precision is already the highest precision supported by a compiler. This is the main reason why iterative refinement is not supported in LINPACK. (The LINPACK manual lists a subroutine that implements mixed precision iterative refinement for single precision data, but it is not part of LINPACK [307, 1979, pp. 1.8–1. 10].) For either form of refinement, a copy of the matrix A needs to be kept in order to form the residual, and this necessitates an extra n^2 elements of storage. A convergence test for terminating the refinement is needed. In addition to revealing when convergence has been achieved, it must signal lack of (sufficiently fast) convergence, which may certainly be experienced when A is very ill conditioned. In the LAPACK driver `xGESVX`, fixed precision iterative refinement is terminated if the componentwise relative backward error $w = w_{|A|,|b|}(\widehat{x}_i)$ satisfies

1. $w \leq u$,
2. w has not decreased by a factor of at least 2 during the current iteration, or
3. five iterations have been performed.

These criteria were chosen to be robust in the face of different BLAS implementations and machine arithmetics. In an implementation of mixed precision iterative refinement it is more natural to test for convergence of the sequence $\{\widehat{x}_i\}$, with a test such as $\|\widehat{x}_i - \widehat{x}_{i-1}\|_\infty / \|\widehat{x}_i\|_\infty \leq u$ (see, e.g., Forsythe and

Moler [396, 1967, p. 65]). However, if A is so ill conditioned that Theorem 11.1 is not applicable, the sequence $\widehat{x}_i$ could converge to a vector other than the solution. This behaviour is very unlikely, and Kahan [626, 1966] quotes a "prominent figure in the world of error-analysis" as saying "Anyone unlucky enough to encounter this sort of calamity has probably already been run over by a truck."

A by-product of extended precision iterative refinement is an estimate of the condition number. Since the error decreases by a factor approximately $h = u\| |A^{-1}||\widehat{L}||\widehat{U}| \|_\infty$, on each iteration (Theorem 11.1), the relative change made to x on the first iteration should be about h, that is, $\|d_1\|_\infty/\|\widehat{x}\|_\infty \approx \eta \approx \kappa_\infty(A)u$. Now that reliable and inexpensive condition estimators are available (Chapter 14) this rough estimate is less important.

An unusual application of iterative refinement is to fault-tolerant computing. Boley et al. [132, 1994] propose solving $Ax = b$ by GEPP or QR factorization, performing one step of fixed precision iterative refinement and then testing whether the a priori residual bound in Theorem 11.4 is satisfied. If the bound is violated then a hardware fault may have occurred and special action is taken.

11.3.1. LAPACK

Iterative refinement is carried out by routines whose names end -RFS, and these routines are called by the expert drivers (name ending -SVX). Iterative refinement is available for all the standard matrix types except triangular matrices, for which the original computed solution already has a componentwise relative backward error of order u. As an example, the expert driver `xGESVX` uses LU factorization with partial pivoting and fixed precision iterative refinement to solve a general system of linear equations with multiple right-hand sides, and the refinement is actually carried out by the routine `xGERFS`.

Problems

11.1. Show that for $A \in \mathbb{R}^{m\times n}$ and $x \in \mathbb{R}^n$, $|A||x| \le \sigma\|A\|_\infty|x|$, where $s = \max_i |x_i|/\min_i |x_i|$.

11.2. Use the analysis of §11.1 to show that, under the conditions of Theorem 11.4, $\|x - \widehat{x}_2\|_\infty/\|x\|_\infty$ is bounded by a multiple of $\mathrm{cond}(A, x)u$ for GEPP after one step of fixed precision iterative refinement.

11.3. Investigate empirically the size of $\| |L||L^{-1}| \|_\infty$ for L from GEPP.

11.4. (Demmel and Higham [291, 1992]) Suppose GEPP with fixed precision iterative refinement is applied to the multiple-right-hand side system $AX = B$, and that refinement of the columns of X is done "in parallel": $R = B - AX$,

$AD = R$, $Y = X + D$. What can be said about the stability of the process if R is computed by conventional multiplication but the second step is done using a fast multiplication technique for which only (12.3) holds?

11.5. (RESEARCH PROBLEM) Is one step of fixed precision iterative refinement sufficient to produce a componentwise relative backward error of order u for Cholesky factorization applied to a symmetric positive definite system $Ax = b$, assuming $\mathrm{cond}(A^{-1})\sigma(A, x)$ is not too large? Answer the same question for the diagonal pivoting method with partial pivoting applied to a symmetric system $Ax = b$.

Chapter 12

Block LU Factorization

> *Block algorithms are advantageous for at least two important reasons.*
> *First, they work with blocks of* data *having* b^2 *elements,*
> *performing* $O(b^3)$ *operations.*
> *The O(b) ratio of work to storage means that*
> *processing elements with an O(b) ratio of*
> *computing speed to input/output bandwidth can be tolerated.*
> *Second, these algorithms are usually rich in matrix multiplication.*
> *This is an advantage because*
> *nearly every modern parallel machine is good at matrix multiplication.*
>
> *—ROBERT* S. SCHREIBER, *Block Algorithms for Parallel Machines (1988)*

> *It should be realized that, with partial pivoting,*
> any matrix *has a triangular factorization.*
> *DECOMP actually works faster when zero pivots occur because they mean that*
> *the corresponding column is already in triangular form.*
>
> — GEORGE E. FORSYTHE, MICHAEL A. MALCOLM, and CLEVE B. MOLER,
> *Computer Methods for Mathematical Computations (1977)*

> *It was quite usual when dealing with very large matrices to*
> *perform an iterative process as follows:*
> *the original matrix would be read from cards and the reduced matrix punched*
> *without more than a single row of the original matrix*
> *being kept in store at any one time;*
> *then the output hopper of the punch would be*
> *transferred to the card reader and the iteration repeated.*
>
> — MARTIN CAMPBELL-KELLY, *Programming the Pi/et ACE (1981)*

12.1. Block Versus Partitioned LU Factorization

As we noted in Chapter 9 (Notes and References), Gaussian elimination (GE) comprises three nested loops that can be ordered in six ways, each yielding a different algorithmic variant of the method. These variants involve different computational kernels: inner product and saxpy operations (level-1 BLAS), or outer product and gaxpy operations (level-2 BLAS). To introduce matrix–matrix operations (level-3 BLAS), which are beneficial for high-performance computing, further manipulation beyond loop reordering is needed. We will use the following terminology, which emphasises an important distinction.

A *partitioned algorithm* is a scalar (or point) algorithm in which the operations have been grouped and reordered into matrix operations.

A *block algorithm* is a generalization of a scalar algorithm in which the basic scalar operations become matrix operations ($a+b$, ab, and a/b become $A+B$, AB, and AB^{-1}), and a matrix property based on the nonzero structure becomes the corresponding property blockwise (in particular, the scalars 0 and 1 become the zero matrix and the identity matrix, respectively). A *block factorization is* defined in a similar way and is usually what a block algorithm computes.

A partitioned version of the outer product form of LU factorization may be developed as follows. For $A \in \mathbb{R}^{n\times n}$ and a given block size r, write

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I_{n-r} \end{bmatrix} \begin{bmatrix} I_r & 0 \\ 0 & B \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & I_{n-r} \end{bmatrix}, \tag{12.1}$$

where A_{11} is $r \times r$. One step of the algorithm consists of factoring $A_{11} = L_{11}U_{11}$, solving the multiple right-hand side triangular systems $L_{11}U_{12} = A_{12}$ and $L_{21}U_{11} = A_{21}$ for U_{12} and L_{21}, respectively, and then forming $B = A_{22} - L_{21}U_{12}$; this procedure is repeated on B. The block operations defining U_{12}, L_{21}, and B are level-3 BLAS operations. This partitioned algorithm does precisely the same arithmetic operations as any other variant of GE, but it does the operations in an order that permits them to be expressed as matrix operations.

A genuine block algorithm computes a *block LU factorization,* which is a factorization $A = LU \in \mathbb{R}^{n\times n}$, where L and U are block triangular and L has identity matrices on the diagonal:

$$L = \begin{bmatrix} I & & & \\ L_{21} & I & & \\ \vdots & & \ddots & \\ L_{m1} & \dots & L_{m,m-1} & I \end{bmatrix}, \quad U = \begin{bmatrix} U_{11} & U_{12} & \dots & U_{1m} \\ & U_{22} & & \vdots \\ & & \ddots & U_{m-1,m} \\ & & & U_{mm} \end{bmatrix}.$$

In general, the blocks can be of different dimensions. Note that this factorization is not the same as a standard LU factorization, because U is not triangular. However, the standard and block LU factorizations are related as follows: if $A = LU$ is a block LU factorization and each U_{ii} has LU factorization $U_{ii} = \overline{L}_{ii}\overline{U}_{ii}$, then $A = L\mathrm{diag}(\overline{L}_{ii}) \cdot \mathrm{diag}(\overline{U}_{ii})U$ is an LU factorization. Conditions for the existence of a block LU factorization are easy to state.

Theorem 12.1. *The matrix* $A = (A_{ij})_{i,j=1}^{m} \in \mathbb{R}^{n\times n}$ *has a unique block LU factorization if and only if the first* $m - 1$ *leading principal block submatrices of* A *are nonsingular.*

Proof. The proof is entirely analogous to the proof of Theorem 9.1. □

This theorem makes clear that a block LU factorization may exist when an LU factorization does not.

If $A_{11} \in \mathbb{R}^{r\times r}$ is nonsingular we can write

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} I & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & S \end{bmatrix}, \tag{12.2}$$

which describes one block step of an outer-product-based algorithm for computing a block LU factorization. Here, $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement of A_{11} in A. If the (1, 1) block of S of appropriate dimension is nonsingular then we can factorize S in a similar manner, and this process can be continued recursively to obtain the complete block LU factorization. The overall algorithm can be expressed as follows.

Algorithm 12.2 (block LU factorization). This algorithm computes a block LU factorization $A = LU \in \mathbb{R}^{n\times n}$, using the notation of (12.2).

```
1. $U_{11} = A_{11}$, $U_{12} = A_{12}$.
2. Solve $L_{21}A_{11} = A_{21}$ for $L_{21}$.
3. $S = A_{22} - L_{21}A_{12}$ (Schur complement).
4. Compute the block LU factorization of $S$, recursively.
```

Given a block LU factorization of A, the solution to a system $Ax = b$ can be obtained by solving $Ly = b$ by forward substitution (since L is triangular) and solving $Ux = y$ by block back substitution. There is freedom in how step 2 of Algorithm 12.2 is accomplished, and how the linear systems with coefficient matrices U_{ii} that arise in the block back substitution are solved. The two main possibilities are as follows.

Implementation 1: A_{11} is factorized by GE with partial pivoting. Step 2 and the solution of linear systems with U_{ii} are accomplished by substitution with the LU factors of A_{11}.

Implementation 2: A_{11}^{-1} is computed explicitly, so that step 2 becomes a matrix multiplication and $Ux = y$ is solved entirely by matrix–vector multiplications. This approach is attractive for parallel machines.

What can be said about the numerical stability of partitioned and block LU factorization? Because the partitioned algorithm is just a rearrangement of standard GE, the standard error analysis applies if the matrix operations are computed in the conventional way. However, if fast matrix multiplication techniques are used (for example, Strassen's method), the standard results are not applicable. Standard results are, in any case, not applicable to block LU factorization; its stability can be very different from that of LU factorization. Therefore we need error analysis for both partitioned and block LU factorization based on general assumptions that permit the use of fast matrix multiplication.

Unless otherwise stated, in this chapter an unsubscripted norm denotes $\|A\| := \max_{i,j} |a_{ij}|$. We make two assumptions about the underlying level-3 BLAS (matrix-matrix operations).

(1) If $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$ then the computed approximation $\widehat{C}$ to $C = AB$ satisfies

$$\widehat{C} = AB + \Delta C, \qquad \|\Delta C\| \le c_1(m,n,p)u\|A\|\,\|B\| + O(u^2), \tag{12.3}$$

where $c_1(m, n, p)$ denotes a constant depending on m, n and p.

(2) The computed solution $\widehat{X}$ to the triangular systems $TX = B$, where $T \in \mathbb{R}^{m\times m}$ and $B \in \mathbb{R}^{m\times p}$, satisfies

$$T\widehat{X} = B + \Delta B, \qquad \|\Delta B\| \le c_2(m,p)u\|T\|\,\|\widehat{X}\| + O(u^2). \tag{12.4}$$

For conventional multiplication and substitution, conditions (12.3) and (12.4) hold with $c_1(m, n, p) = n^2$ and $c_2(m,p) = m^2$. For implementations based on Strassen's method, (12.3) and (12.4) hold with c_1 and c_2 rather complicated functions of the dimensions m, n, p and the threshold no that determines the level of recursion (see Theorem 22.2 and [544, 1990]).

12.2. Error Analysis of Partitioned LU Factorization

An error analysis for partitioned LU factorization must answer two questions. The first is whether partitioned LU factorization becomes unstable in some fundamental way when fast matrix multiplication is used. The second is whether the constants in (12.3) and (12.4) are propagated stably into the final error bound (exponential growth of the constants would be disastrous).

We will assume that the block level LU factorization is done in such a way that the computed LU factors of $A_{11} \in \mathbb{R}^{r\times r}$ satisfy

$$\widehat{L}_{11}\widehat{U}_{11} = A_{11} + \Delta A_{11}, \quad \|\Delta A_{11}\| \le c_3(r)u\|\widehat{L}_{11}\|\,\|\widehat{U}_{11}\| + O(u^2). \tag{12.5}$$

Theorem 12.3 (Demmel and Higham). *Under the assumptions* (12.3), (12.4), *and* (12.5), *the LU factors of* $A \in \mathbb{R}^{n\times n}$ *computed using the partitioned* *outer product form of LU factorization with block size r satisfiy* $\widehat{L}\widehat{U}$ = *A*+ *DA*, *where*

$$\|\Delta A\| \le u\big(\delta(n,r)\|A\| + \theta(n,r)\|\widehat{L}\|\,\|\widehat{U}\|\big) + O(u^2), \qquad (12.6)$$

and where

$$\begin{aligned}\delta(n,r) &= 1 + \delta(n-r,r), \quad \delta(r,r) = 0,\\ \theta(n,r) &= \max\{c_3(r), c_2(r,n-r), 1 + c_1(n-r,r,n-r) + \delta(n-r,r)\\ &\qquad + \theta(n-r,r)\}, \qquad \theta(r,r) = c_3(r).\end{aligned}$$

Proof. The proof is essentially inductive. To save clutter we will omit "$+O(u^2)$" from each bound. For $n = r$, the result holds trivially. Consider the first block stage of the factorization, with the partitioning (12.1). The assumptions imply that

$$\widehat{L}_{11}\widehat{U}_{12} = A_{12} + \Delta A_{12}, \qquad \|\Delta A_{12}\| \le c_2(r,n-r)u\|\widehat{L}_{11}\|\,\|\widehat{U}_{12}\|, \quad (12.7)$$

$$\widehat{L}_{21}\widehat{U}_{11} = A_{21} + \Delta A_{21}, \qquad \|\Delta A_{21}\| \le c_2(r,n-r)u\|\widehat{L}_{21}\|\,\|\widehat{U}_{11}\|. \quad (12.8)$$

To obtain $B = A_{22} - L_{21}U_{12}$ we first compute $C = \widehat{L}_{21}\widehat{U}_{12}$, obtaining

$$\widehat{C} = \widehat{L}_{21}\widehat{U}_{12} + \Delta C, \qquad \|\Delta C\| \le c_1(n-r,r,n-r)u\|\widehat{L}_{21}\|\,\|\widehat{U}_{12}\|,$$

and then subtract from A_{22}, obtaining

$$\widehat{B} = A_{22} - \widehat{C} + F, \qquad \|F\| \le u(\|A_{22}\| + \|\widehat{C}\|). \qquad (12.9)$$

It follows that

$$\widehat{B} = A_{22} - \widehat{L}_{21}\widehat{U}_{12} + \Delta B, \qquad (12.10a)$$

$$\begin{aligned}\|\Delta B\| \le u\big(&\|A_{22}\| + \|\widehat{L}_{21}\|\,\|\widehat{U}_{12}\|\\ &+ c_1(n-r,r,n-r)\|\widehat{L}_{21}\|\,\|\widehat{U}_{12}\|\big). \qquad (12.10b)\end{aligned}$$

The remainder of the algorithm consists of the computation of the LU factorization of *B*, and by our inductive assumption (12.6), the computed LU factors satisfy

$$\widehat{L}_{22}\widehat{U}_{22} = \widehat{B} + \Delta\widehat{B}, \qquad (12.11a)$$

$$\|\Delta\widehat{B}\| \le \delta(n-r,r)u\|\widehat{B}\| + \theta(n-r,r)u\|\widehat{L}_{22}\|\,\|\widehat{U}_{22}\|. \qquad (12.11b)$$

Combining (12.10) and (12.11), and bounding $\|\widehat{B}\|$ using (12.9), we obtain

$$\widehat{L}_{21}\widehat{U}_{12} + \widehat{L}_{22}\widehat{U}_{22} = A_{22} + \Delta A_{22},$$

$$\begin{aligned}\Delta A_{22}\| \le u\big([1 + \delta(n-r,r)]\|A_{22}\| &+ [1 + c_1(n-r,r,n-r) + \delta(n-r,r)]\\ \times \|\widehat{L}_{21}\|\,\|\widehat{U}_{12}\| &+ \theta(n-r,r)\|\widehat{L}_{22}\|\,\|\widehat{U}_{22}\|\big). \qquad (12.12)\end{aligned}$$

Collecting (12.5), (12.7), (12.8), and (12.12) we have $\widehat{L}\widehat{U} = A + \Delta A$, where bounds on $\|\Delta A_{ij}\|$ are given in the equations just mentioned. These bounds for the blocks of ΔA can be weakened slightly and expressed together in the more succinct form (12.6). □

These recurrences for $\delta(n,r)$ and $q(n,r)$ show that the basic error constants in assumptions (12.3), (12.4), and (12.5) combine additively at worst. Thus, the backward error analysis for the LU factorization is commensurate with the error analysis for the particular implementation of the BLAS3 employed in the partitioned factorization. In the case of the conventional BLAS3 we obtain a Wilkinson-style result for GE without pivoting, with $q(n,r) = O(n^3)$ (the growth factor is hidden in $\widehat{L}$ and $\widehat{U}$).

Although the above analysis is phrased in terms of the partitioned outer product form of LU factorization, the same result holds for other "*ijk*" partitioned forms (with slightly different constants), for example, the gaxpy or sdot forms. There is no difficulty in extending the analysis to cover partial pivoting and solution of $Ax = b$ using the computed LU factorization (see Problem 12.6).

12.3. Error Analysis of Block LU Factorization

Now we turn to block LU factorization. We assume that the computed matrices L_{21} from step 2 of Algorithm 12.2 satisfy

$$\widehat{L}_{21}A_{11} = A_{21} + E_{21}, \qquad \|E_{21}\| \le c_4(n,r)u\|\widehat{L}_{21}\|\,\|A_{11}\| + O(u^2). \tag{12.13}$$

We also assume that when a system $U_{ii}x_i = d_i$ of order r is solved, the computed solution $\widehat{x}_i$ satisfies

$$(U_{ii} + \Delta U_{ii})\widehat{x}_i = d_i, \qquad \|\Delta U_{ii}\| \le c_5(r)u\|U_{ii}\| + O(u^2). \tag{12.14}$$

The assumptions (12.13) and (12.14) are satisfied for Implementation 1 of Algorithm 12.2 and are sufficient to prove the following result.

Theorem 12.4 (Demmel, Higham, and Schreiber). *Let* $\widehat{L}$ *and* $\widehat{U}$ *be the computed block LU factors of* $A \in \mathbb{R}^{n\times n}$ *from Algorithm* 12.2 *(with Implementation* 1*), and let* $\widehat{x}$ *be the computed solution to* $Ax = b$. *Under the assumptions* (12.3), (12.13), *and* (12.14),

$$\begin{gathered}\widehat{L}\widehat{U} = A + \Delta A_1, \qquad (A + \Delta A_2)\widehat{x} = b,\\ \|\Delta A_i\| \le d_n u\big(\|A\| + \|\widehat{L}\|\,\|\widehat{U}\|\big) + O(u^2), \quad i = 1{:}2,\end{gathered} \tag{12.15}$$

where the constant d_n *is commensurate with those in the assumptions.*

Proof. We omit the proof (see Demmel, Higham, and Schreiber [293, 1995] for details). It is similar to the proof of Theorem 12.3. □

The bounds in Theorem 12.4 are valid also for other versions of block LU factorization obtained by "block loop reordering", such as a block gaxpy based algorithm.

Theorem 12.4 shows that the stability of block LU factorization is determined by the ratio $\||L||U|\|/\|A\|$ (numerical experiments show that the bounds are, in fact, reasonably sharp). If this ratio is bounded by a modest function of n, then L and U are the true factors of a matrix close to A, and $\widehat{x}$ solves a slightly perturbed system. However, $\|\widehat{L}\|\,\|\widehat{U}\|$ can exceed $\|A\|$ by an arbitrary factor, even if A is symmetric positive definite or diagonally dominant by rows. Indeed, $\|L\| \geq \|L_{21}\| = \|A_{21}A_{11}^{-1}\|$, using the partitioning (12.2), and this lower bound for $\|L\|$ can be arbitrarily large. In the following two subsections we investigate this instability more closely and show that $\||L||U|\|$ can be bounded in a useful way for particular classes of A. Without further comment we make the reasonable assumption that $\||L||U|\| \approx \|L\|\|U\|$, so that these bounds maybe used in Theorem 12.4.

What can be said for Implementation 2? Suppose, for simplicity, that the inverses A_{11}^{-1} (which are used in step 2 of Algorithm 12.2 and in the block back substitution) are computed exactly. Then the best bounds of the forms (12.13) and (12.14) are

$$\begin{aligned} \widehat{L}_{21}A_{11} &= A_{21} + \Delta A_{21}, && \|\Delta A_{21}\| \leq c_4(n,r)u\kappa(A_{11})\|A_{21}\| + O(u^2),\\ (U_{ii} + \Delta U_{ii})\widehat{x}_i &= d_i, && \|\Delta U_{ii}\| \leq c_5(r)u\kappa(U_{ii})\|U_{ii}\| + O(u^2). \end{aligned}$$

Working from these results, we find that Theorem 12.4 still holds provided the first-order terms in the bounds in (12.15) are multiplied by $\max_i \kappa(U_{ii})$. This suggests that Implementation 2 of Algorithm 12.2 can be much less stable than Implementation 1 when the diagonal blocks of U are ill conditioned, and this is confirmed by numerical experiments.

12.3.1. Block Diagonal Dominance

One class of matrices for which block LU factorization has long been known to be stable is block tridiagonal matrices that are diagonally dominant in an appropriate block sense. A general matrix $A \in \mathbb{R}^{n\times n}$ is *block diagonally dominant by columns* with respect to a given partitioning $A = (A_{ij})$ and a given norm if, for all j,

$$\|A_{jj}^{-1}\|^{-1} - \sum_{i\neq j}\|A_{ij}\| = \gamma_j \geq 0. \qquad (12.16)$$

A is block diagonally dominant by rows if A^T is block diagonally dominant by columns. For the block size 1, the usual property of point diagonal dominance is obtained. Note that for the 1- and co-norms diagonal dominance does not imply block diagonal dominance, nor does the reverse implication hold (see Problem 12.2). Throughout our analysis of block diagonal dominance we take the norm to be an arbitrary subordinate matrix norm.

First, we show that for block diagonally dominant matrices a block LU factorization exists, using the key property that block diagonal dominance is inherited by the Schur complements obtained in the course of the factorization. In the analysis we assume that A has m block rows and columns.

Theorem 12.5 (Demmel, Higham, and Schreiber). *Suppose* $A \in \mathbb{R}^{n\times n}$ *is nonsingular and block diagonally dominant by rows or columns with respect to a subordinate matrix norm in* (12.16). *Then A has a block LU factorization, and all the Schur complements arising in Algorithm* 12.2 *have the same kind of diagonal dominance as A.*

Proof. This proof is a generalization of Wilkinson's proof of the corresponding result for point diagonally dominant matrices [1085, 1961, pp. 288–289], [470, 1989, p. 120] (as is the proof of Theorem 12.6 below). We consider the case of block diagonal dominance by columns; the proof for row-wise diagonal dominance is analogous.

The first step of Algorithm 12.2 succeeds, since A_{11} is nonsingular, producing a matrix that we can write as

$$A^{(2)} = \begin{bmatrix} U_{11} & U_{12} \\ 0 & S \end{bmatrix}.$$

For $j = 2{:}m$ we have

$$\begin{aligned}
\sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{ij}^{(2)}\| &= \sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{ij} - A_{i1}A_{11}^{-1}A_{1j}\| \\
&\le \sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{ij}\| + \|A_{1j}\|\,\|A_{11}^{-1}\| \sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{i1}\| \\
&\le \sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{ij}\| + \|A_{1j}\|\,\|A_{11}^{-1}\| \big(\|A_{11}^{-1}\|^{-1} - \|A_{j1}\|\big), \quad \text{using (12.16)} \\
&= \sum_{\substack{i=2\\ i\neq j}}^{m} \|A_{ij}\| + \|A_{1j}\| - \|A_{1j}\|\,\|A_{11}^{-1}\|\,\|A_{j1}\| \\
&\le \|A_{jj}^{-1}\|^{-1} - \|A_{1j}\|\,\|A_{11}^{-1}\|\,\|A_{j1}\|, \quad \text{using (12.16),} \\
&= \min_{\|x\|=1} \|A_{jj}x\| - \|A_{1j}\|\,\|A_{11}^{-1}\|\,\|A_{j1}\|
\end{aligned}$$

$$\begin{aligned} &\le \min_{\|x\|=1} \|(A_{jj} - A_{j1}A_{11}^{-1}A_{1j})x\| \\ &= \min_{\|x\|=1} \|A_{jj}^{(2)}x\|. \end{aligned} \qquad (12.17)$$

Now if $A_{jj}^{(2)}$ is singular it follows that $\sum_{i=2,i\ne j}^{m} \|A_{ij}^{(2)}\| = 0$; therefore $A^{(2)}$, and hence also A, is singular, which is a contradiction. Thus $A_{jj}^{(2)}$ is nonsingular, and (12.17) can be rewritten

$$\sum_{\substack{i=2\\ i\ne j}}^{m} \|A_{ij}^{(2)}\| \le \|A_{jj}^{(2)^{-1}}\|^{-1},$$

showing that $A^{(2)}$ is block diagonally dominant by columns. The result follows by induction. $\square$

The next result allows us to bound $\|U\|$ for a block diagonally dominant matrix.

Theorem 12.6 (Demmel, Higham, and Schreiber). *Let A satisfy the conditions of Theorem* 12.5. *If $A^{(k)}$ denotes the matrix obtained after $k-1$ steps of Algorithm* 12.2, *then*

$$\max_{k\le i,j\le m} \|A_{ij}^{(k)}\| \le 2 \max_{1\le i,j\le m} \|A_{ij}\|.$$

Proof. Let A be block diagonally dominant by columns (the proof for row diagonal dominance is similar). Then

$$\begin{aligned} \sum_{i=2}^{m} \|A_{ij}^{(2)}\| &= \sum_{i=2}^{m} \|A_{ij} - A_{i1}A_{11}^{-1}A_{1j}\| \\ &\le \sum_{i=2}^{m} \|A_{ij}\| + \|A_{1j}\|\,\|A_{11}^{-1}\| \sum_{i=2}^{m} \|A_{i1}\| \\ &\le \sum_{i=1}^{m} \|A_{ij}\|, \end{aligned}$$

using (12.16). By induction, using Theorem 12.5, it follows that $\sum_{i=k}^{m} \|A_{ij}^{(k)}\| \le \sum_{i=1}^{m} \|A_{ij}\|$. This yields

$$\max_{k\le i,j\le m} \|A_{ij}^{(k)}\| \le \max_{k\le j\le m} \sum_{i=k}^{m} \|A_{ij}^{(k)}\| \le \max_{k\le j\le m} \sum_{i=1}^{m} \|A_{ij}\|.$$

From (12.16), $\sum_{i\ne j} \|A_{ij}\| \le \|A_{jj}^{-1}\|^{-1} \le \|A_{jj}\|$, so

$$\max_{k\le i,j\le m} \|A_{ij}^{(k)}\| \le 2 \max_{k\le j\le m} \|A_{jj}\| \le 2 \max_{1\le j\le m} \|A_{jj}\| = 2 \max_{1\le i,j\le m} \|A_{ij}\|. \qquad \square$$

The implications of Theorems 12.5 and 12.6 for stability are as follows. Suppose A is block diagonally dominant by columns. Also, assume for the moment that the (subordinate) norm has the property that

$$\max_{i,j} \|A_{ij}\| \le \|A\| \le \sum_{i,j} \|A_{ij}\|, \tag{12.18}$$

which holds for any p-norm, for example. The subdiagonal blocks in the first block column of L are given by $L_{i1} = A_{i1}A_{11}^{-1}$ and so $\|[L_{21}^T, \dots, L_{m1}^T]^T\| \le 1$, by (12.16) and (12.18). From Theorem 12.5 it follows that $\|[L_{j+1,j}^T, \dots, L_{mj}^T]^T\| \le 1$ for $j = 2{:}m$. Since $U_{ij} = A_{ij}^{(i)}$ for $j \ge i$, Theorem 12.6 shows that $\|U_{ij}\| \le 2\|A\|$ for each block of U (and $\|U_{ij}\| \le \|A\|$). Therefore $\|L\| \le m$ and $\|U\| \le m^2\|A\|$, and so $\|L\|\|U\| \le m^3\|A\|$. For particular norms the bounds on the blocks of L and U yield a smaller bound for $\|L\|$ and $\|U\|$. For example, for the 1-norm we have $\|L\|_1\|U\|_1 \le 2m\|A\|_1$ and for the ∞-norm $\|L\|_\infty\|U\|_\infty \le 2m^2\|A\|_\infty$. We conclude that block LU factorization is stable if A is block diagonally dominant by columns with respect to any subordinate matrix norm satisfying (12.18).

Unfortunately, block LU factorization can be unstable when A is block diagonally dominant by rows, for although Theorem 12.6 guarantees that $\|U_{ij}\| \le 2\|A\|$, $\|L\|$ can be arbitrarily large. This can be seen from the example

$$A = \begin{bmatrix} A_{11} & 0 \\ \frac{1}{2}I & I \end{bmatrix} = \begin{bmatrix} I & 0 \\ \frac{1}{2}A_{11}^{-1} & I \end{bmatrix} \begin{bmatrix} A_{11} & 0 \\ 0 & I \end{bmatrix} = LU,$$

where A is block diagonally dominant by rows in any subordinate norm for any nonsingular matrix A_{11}. It is easy to confirm numerically that block LU factorization can be unstable on matrices of this form.

Next, we bound $\|L\|\|U\|$ for a general matrix and then specialize to point diagonal dominance. *From this point on we use the norm* $\|A\| := \max_{i,j} |a_{ij}|$. We partition A according to

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{r\times r}, \tag{12.19}$$

and denote by ρ_n the growth factor for GE without pivoting. We assume that GE applied to A succeeds.

To bound $\|L\|$, we note that, under the partitioning (12.19), for the first block stage of Algorithm 12.2 we have $\|L_{21}\| = \|A_{21}A_{11}^{-1}\| \le n\rho_n\kappa(A)$ (see Problem 12.4). Since the algorithm works recursively with the Schur complement S, and since every Schur complement satisfies $\kappa(S) \le \rho_n\kappa(A)$ (see Problem 12.4), each subsequently computed subdiagonal block of L has norm at most $n\rho_n^2\kappa(A)$ Since U is composed of elements of A together with elements of Schur complements of A,

$$\|U\| \le \rho_n\|A\|. \tag{12.20}$$

Overall, then, for a general matrix $A \in \mathbb{R}^{n\times n}$,

$$\|L\|\,\|U\| \le n\rho_n^2\kappa(A)\cdot\rho_n\|A\| = n\rho_n^3\kappa(A)\|A\|. \qquad (12.21)$$

Thus, block LU factorization is stable for a general matrix A as long as GE is stable for A (that is, ρ_n is of order 1) and A is well conditioned.

If A is point diagonally dominant by columns then, since every Schur complement enjoys the same property, we have $\|L_{ij}\| \le 1$ for $i > j$, by Problem 12.5. Hence $\|L\| = 1$. Furthermore, $\rho_n \le 2$ (Theorem 9.8 or Theorem 12.6), giving $\|U\| \le 2\|A\|$ by (12.20), and so

$$\|L\|\|U\| \le 2\|A\|.$$

Thus block LU factorization is perfectly stable for a matrix point diagonally dominant by columns.

If A is point diagonally dominant by rows then the best we can do is to take $\rho_n \le 2$ in (12.21), obtaining

$$\|L\|\|U\| \le 8nk(A)\|A\|. \qquad (12.22)$$

Hence for point row diagonally dominant matrices, stability is guaranteed if A is well conditioned. This in turn is guaranteed if the row diagonal dominance amounts γ_j in the analogue of (12.16) for point row diagonal dominance are sufficiently large relative to $\|A\|$, because $\|A^{-1}\|_\infty \le (\min_j \gamma_j)^{-1}$ (see problem 8.7(a)).

12.3.2. Symmetric Positive Definite Matrices

Further useful results about the stability of block LU factorization can be derived for symmetric positive definite matrices. First, note that the existence of a block LU factorization is immediate for such matrices, since all their leading principal submatrices are nonsingular. Let A be a symmetric positive definite matrix, partitioned as

$$A = \begin{bmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{r\times r}.$$

The definiteness implies certain relations among the submatrices A_{ij} that can be used to obtain a stronger bound for $\|L\|_2$ than can be deduced for a general matrix (cf. Problem 12.4).

Lemma 12.7. *If A is symmetric positive definite then* $\|A_{21}A_{11}^{-1}\|_2 \le \kappa_2(A)^{1/2}$.

Proof. This lemma is a corollary of Lemma 10.12, but we give a separate proof. Let A have the Cholesky factorization

$$A = \begin{bmatrix} R_{11}^T & 0 \\ R_{12}^T & R_{22}^T \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix}, \qquad R_{11} \in \mathbb{R}^{r\times r}.$$

Table 12.1. *Stability of block and point L U factorization.* p_n *is the growth factor for GE without pivoting.*

Matrix property	Block LU	Point LU
Symmetric positive definite	$\kappa(A)^{1/2}$	1
Block column diagonally dominant	1	p_n
Point column diagonally dominant	1	1
Block row diagonally dominant	$\rho_n^3\kappa(A)$	p_n
Point row diagonally dominant	$\kappa(A)$	1
Arbitrary	$\rho_n^3\kappa(A)$	p_n

Then $A_{21}A_{11}^{-1} = R_{12}^TR_{11}\cdot R_{11}^{-1}R_{11}^{-T} = R_{12}^TR_{11}^{-T}$, so

$$\|A_{21}A_{11}^{-1}\|_2 \le \|R_{12}\|_2\|R_{11}^{-1}\|_2 \le \|R\|_2\|R^{-1}\|_2 = \kappa_2(R) = \kappa_2(A)^{1/2}. \qquad \square$$

The following lemma is proved in a way similar to the second inequality in Problem 12.4.

Lemma 12.8. *If A is symmetric positive definite then the Schur complement* $S = A_{22} - {}_{21}A_{11}^{-1}A_{21}^T$ *satisfies* $\kappa_2(S) \le \kappa_2(A)$.

Using the same reasoning as in the last subsection, we deduce from these two lemmas that each subdiagonal block of L is bounded in 2-norm by $\kappa_2(A)^{1/2}$. Therefore $\|L\|_2 \le 1 + m\kappa_2(A)^{1/2}$, where there are m block stages in the algorithm. Also, it can be shown that $\|U\|_2 \le \sqrt{m}\|A\|_2$. Hence

$$\|L\|_2\|U\|_2 \le \sqrt{m}(1 + m\kappa_2(A)^{1/2})\|A\|_2. \tag{12.23}$$

It follows from Theorem 12.4 that when Algorithm 12.2 is applied to a symmetric positive definite matrix A, the backward errors for the LU factorization and the subsequent solution of a linear system are both bounded by

$$c_n\sqrt{m}u\|A\|_2(2 + m\kappa_2(A)^{1/2}) + O(u^2). \tag{12.24}$$

Any resulting bound for $\|x-\widehat{x}\|_2/\|x\|_2$ will be proportional to $\kappa_2(A)^{3/2}$, rather than $\kappa_2(A)$ as for a stable method. This suggests that block LU factorization can lose up to 50% more digits of accuracy in x than a stable method for solving symmetric positive definite linear systems. The positive conclusion to be drawn, however, is that block LU factorization is guaranteed to be stable for a symmetric positive definite matrix that is well conditioned.

The stability results for block LU factorization are summarized in Table 12.1, which tabulates a bound for $\|A - LU\|/(c_nu\|A\|)$ for block and point

LU factorization for the matrix properties considered in this chapter. The constant c_n incorporates any constants in the bound that depend polynomially on the dimension, so a value of 1 in the table indicates unconditional stability.

12.4. Notes and References

The distinction between a partitioned algorithm and a block algorithm is rarely made in the literature (exceptions include the papers by Schreiber [902, 1988] and Demmel, Higham, and Schreiber [293, 1995]); the term "block algorithm" is frequently used to describe both types of algorithm. A partitioned algorithm might also be called a "blocked algorithm" (as is done by Dongarra, Duff, Sorensen, and van der Vorst [315, 1991]), but the similarity of this term to "block algorithm" can cause confusion and so we do not recommend this terminology. Note that in the particular case of matrix multiplication, partitioned and block algorithms are equivalent. Our treatment of partitioned LU factorization has focused on the stability aspects; for further details, particularly concerning implementation on high-performance computers, see Dongarra, Duff, Sorensen, and van der Vorst [315, 1991] and Golub and Van Loan [470, 1989].

Block LU factorization appears to have first been proposed for block tridiagonal matrices, which frequently arise in the discretization of partial differential equations. References relevant to this application include Isaacson and Keller [607, 1966, p. 59], Varah [1048, 1972], Bank and Rose [53, 1977], Mattheij [737, 1984], [738, 1984], and Concus, Golub, and Meurant [235, 1985].

For an application of block LU factorization to linear programming, see Eldersveld and Saunders [351, 1992].

Theorem 12.3 is from Demmel and Higham [291, 1992]. The results in §12.3 are from Demmel, Higham, and Schreiber [293, 1995], which extends earlier analysis of block LU factorization by Demmel and Higham [291, 1992].

Block diagonal dominance was introduced by Feingold and Varga [366, 1962], and has been used mainly in generalizations of the Gershgorin circle theorem. Varah [1048, 1972] obtained bounds on $||L||$ and $||U||$ for block diagonally dominant block tridiagonal matrices; see Problem 12.1.

Theorem 12.5 is obtained in the case of block diagonal dominance by rows with $\min_j \gamma_j > 0$ by Polman [837, 1987]; the proof in [837, 1987] makes use of the corresponding result for point diagonal dominance and thus differs from the proof we have given.

At the cost of a much more difficult proof, Lemma 12.7 can be strengthened to the attainable bound $\|A_{21}A_{11}^{-1}\|_2 \le (\kappa_2(A)^{1/2} - \kappa_2(A)^{-1/2})/2$, as shown by Demmel [279, 1983, Thin. 4], but the weaker bound is sufficient for our purposes.

12.4.1. LAPACK

LAPACK does not implement block LU factorization, but its LU factorization (and related) routines for full matrices employ partitioned LU factorization in order to exploit the level-3 BLAS and thereby to be efficient on high-performance machines.

Problems

12.1. (Varah [1048, 1972]) Suppose A is block tridiagonal and has the block LU factorization $A = LU$ (so that L and U are block bidiagonal and $U_{i,i+1} = A_{i,i+1}$). Show that if A is block diagonally dominant by columns then

$$\|L_{i,i-1}\| \le 1, \qquad \|U_{ii}\| \le \|A_{ii}\| + \|A_{i-1,i}\|,$$

while if A is block diagonally dominant by rows then

$$\|L_{i,i-1}\| \le \|A_{i,i-1}\|/\|A_{i-1,i}\|, \qquad \|U_{ii}\| \le \|A_{ii}\| + \|A_{i,i-1}\|.$$

What can be deduced about the stability of the factorization and has the block classes of matrices?

12.2. Show that for the 1- and co-norms diagonal dominance does not imply block diagonal dominance, and vice versa.

12.3. If $A \in \mathbb{R}^{n\times n}$ is symmetric, has positive diagonal elements, and is block diagonally dominant by rows, must it be positive definite?

12.4. Let $A \in \mathbb{R}^{n\times n}$ be partitioned

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \qquad A_{11} \in \mathbb{R}^{r\times r}, \tag{12.25}$$

with A_{11} nonsingular. Let $\|A\| := \max_{ij} |a_{ij}|$. Show that $\|A_{21}A_{11}^{-1}\| \le n p_n \kappa(A)$. Show that the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$ satisfies $\kappa(S) \le p_n \kappa(A)$.

12.5. Let $A \in \mathbb{R}^{n\times n}$ be partitioned as in (12.25), with A_{11} nonsingular, and suppose that A is point diagonally dominant by columns. Show that $\|A_{21}A_{11}^{-1}\|_1 \le 1$.

12.6. Show that under the conditions of Theorem 12.3 the computed solution to $Ax = b$ satisfies

$$(A + \Delta A)\widehat{x} = b, \qquad \|\Delta A\| \le c_n u\bigl(\|A\| + \|\widehat{L}\|\,\|\widehat{U}\|\bigr) + O(u^2),$$

and the computed solution to the multiple right-hand side system $AX = B$ (where (12.4) is assumed to hold for the multiple right-hand side triangular solves) satisfies

$$\|A\widehat{X} - B\| \le c_n u\big(\|A\| + \|\widehat{L}\|\,\|\widehat{U}\|\big)\,\|\widehat{X}\| + O(u^2).$$

In both cases, c_n is a constant depending on n and the block size.

12.7. Let $X = \left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right] \in \mathbb{R}^{n\times n}$, where A is square and nonsingular. Show that

$$\det(X) = \det(A)\det(D - CA^{-1}B).$$

Assuming A, B, C, D are all $m \times m$, give a condition under which $\det(X) = \det(AD - CB)$.

Chapter 13

Matrix Inversion

It is amusing to remark that we were so involved with
matrix inversion that we probably talked of nothing else for months.
Just in this period Mrs. von Neumann acquired a big,
rather wild but gentle Irish Setter puppy,
which she called inverse in honor of our work!

— HERMAN H. GOLDSTINE, *The Computer.'*
From Pascal to von Neumann (1972)

The most computationally intensive portion
of the tasks assigned to the processors is
integrating the KKR matrix inverse over the first Brillouin zone.
To evaluate the integral,
hundreds or possibly thousands of complex double precision matrices
of order between 80 and 300 must be formed and inverted.
Each matrix corresponds to a different vertex of the tetrahedrons
into which the Brillouin zone has been subdivided.

— M. T. HEATH, G. A. GEIST, and J. B. DRAKE, *Superconductivity*
in Early Experience with the Intel iPSC/860
at Oak Ridge National Laboratory (1990)

Press $\boxed{1/x}$ *to invert the matrix.*
Note that matrix inversion can produce erroneous results
if you are using iii-conditioned matrices.

— HEWLETT-PACKARD, *HP 48G Series User's Guide (1993)*

Almost anything you can do with A^{-1} *can be done without* it.

— GEORGE E. FORSYTHE and CLEVE B. MOLER,
Computer Solution of Linear Algebraic Systems (1967)

13.1. Use and Abuse of the Matrix Inverse

To most numerical analysts, matrix inversion is a sin. Forsythe, Malcolm, and Moler put it well when they say [395, 1977, p. 31] "In the vast majority of practical computational problems, it is unnecessary and inadvisable to actually compute A^{-1}." The best example of a problem in which the matrix inverse should not be computed is the linear equations problem $Ax = b$. Computing the solution as $x = A^{-1} \times b$ requires $2n^3$ flops, assuming A^{-1} is computed by Gaussian elimination with partial pivoting (GEPP), whereas GEPP applied directly to the system costs only $2n^3/3$ flops.

Not only is the inversion approach three times more expensive, but it is much less stable. Suppose $X = A^{-1}$ is formed *exactly*, and that the only rounding errors are in forming $x = fl(Xb)$. Then $\widehat{x} = (X + \Delta X)b$, where $|\Delta X| \le \gamma_n|X|$, by (3.10). So $A\widehat{x} = A(X + \Delta X)b = (I + A\Delta X)b$, and the best possible residual bound is

$$|b - A\widehat{x}| \le \gamma_n |A||A^{-1}||b|.$$

For GEPP, Theorem 9.4 yields

$$|b - A\widehat{x}| \le 2\gamma_n |\widehat{L}||\widehat{U}||\widehat{x}|.$$

Since it is usually true that $\| |\widehat{L}||\widehat{U}| \|_\infty \approx \|A\|_\infty$ for GEPP, we see that the matrix inversion approach is likely to give a much larger residual than GEPP if A is ill conditioned and if $\|x\|_\infty \ll \| |A^{-1}||b| \|_\infty$. For example, we solved 50 25×25 systems $Ax = b$ in MATLAB, where the elements of x are taken from the normal $N(0, 1)$ distribution and A is random with $\kappa_2(A) = u^{-1/2} \approx 9 \times 10^7$. As Table 13.1 shows, the inversion approach provided much larger backward errors than GEPP in this experiment.

Given the inexpedience of matrix inversion, why devote a chapter to it? The answer is twofold. First, there *are* situations in which a matrix inverse must be computed. Examples are in statistics [54, 1974, §7.5], [721, 1984, §2.3], [744, 1989, p. 342 ff], where the inverse can convey important statistical information, in certain matrix iterations arising in eigenvalue-related problems [37, 1993], [174, 1987], [566, 1994], and in numerical integrations arising in

Table 13.1. *Backward errors $\eta_{A,b}(\widehat{x})$ for the ∞-norm.*

	min	max
$x = A^{-1} \times b$	6.66e-12	1.69e-10
GEPP	3.44e-18	7.56e-17

superconductivity computations [509, 1990] (see the quotation at the start of the chapter). Second, methods for matrix inversion display a wide variety of stability properties, making for instructive and challenging error analysis. (Indeed, the first major rounding error analysis to be published, that of von Neumann and Goldstine, was for matrix inversion-see §9.6).

Matrix inversion can be done in many different ways—in fact, there are more computationally distinct possibilities than for any other basic matrix computation. For example, in triangular matrix inversion different loop orderings are possible and either triangular matrix–vector multiplication, solution of a triangular system, or a rank-1 update of a rectangular matrix can be employed inside the outer loop. More generally, given a factorization $PA = LU$, two ways to evaluate A^{-1} are. as $A^{-1} = U^{-1} \times L^{-1} \times P$, and as the solution to $UA^{-1} = L^{-1} \times P$. These methods generally achieve different levels of efficiency on high-performance computers, and they propagate rounding errors in different ways. We concentrate in this chapter on the numerical stability, but comment briefly on performance issues.

The quality of an approximation $Y \approx A^{-1}$ can be assessed by looking at the right and left residuals, $AY — I$ and $YA — I$, and the forward error, $Y - A^{-1}$. Suppose we perturb $A \rightarrow A + DA$ with $|DA| \le \epsilon|A|$; thus, we are making relative perturbations of size at most ϵ to the elements of A. If $Y = (A + DA)^{-1}$ then $(A + DA)Y = Y(A + DA) = I$, so that

(13.1)

(13.2)

and, since $(A + DA)^{-1} = A^{-1} - A^{-1}DAA^{-1} + \mathcal{O}(\epsilon^2)$,

$$|A^{-1} - Y| \le \epsilon|A^{-1}||A||A^{-1}| + O(\epsilon^2). \tag{13.3}$$

(Note that (13.3) can also be derived from (13.1) or (13.2).) The bounds (13.1)-(13.3) represent "ideal" bounds for a computed approximation Y to A^{-1}, if we regard ϵ as a small multiple of the unit roundoff u. We will show that, for triangular matrix inversion, appropriate methods do indeed achieve (13.1) or (13.2) (but not both) and (13.3).

It is important to note that neither (13.1), (13.2), nor (13.3) implies that $Y + DY = (A + DA)^{-1}$ with $\|\Delta A\|_\infty \le \epsilon\|A\|_\infty$ and $\|\Delta Y\|_\infty \le \epsilon\|Y\|_\infty$, that is, Y need not be close to the inverse of a matrix near to A, even in the norm sense. Indeed, such a result would imply that both the left and right residuals are bounded in norm by $(2\epsilon + \epsilon^2)\|A\|_\infty\|Y\|_\infty$, and this is not the case for any of the methods we will consider.

To illustrate the latter point we give a numerical example. Define the matrix $A_n \in \mathbb{R}^{n\times n}$ as triu(qr(vand(n))), in MATLAB notation (vand is a routine from the Test Matrix Toolbox—see Appendix E); in other words, A_n is the upper triangular QR factor of the $n \times n$ Vandermonde matrix based on

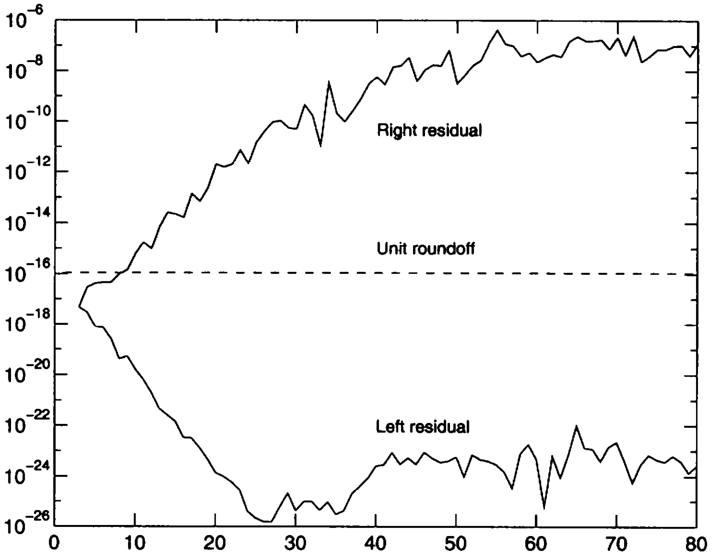


Figure 13.1. *Residuals for inverses computed by* MATLAB'S **INV** *function.*

equispaced points on [0, 1]. We inverted A_n, for $n = 1{:}80$, using MATLAB'S **INV** function, which uses GEPP. The left and right normwise relative residuals

$$\mathrm{res_L} = \frac{\|\widehat{X}A - I\|_\infty}{\|\widehat{X}\|_\infty \|A\|_\infty}, \qquad \mathrm{res_R} = \frac{\|A\widehat{X} - I\|_\infty}{\|A\|_\infty \|\widehat{X}\|_\infty},$$

are plotted in Figure 13.1. We see that while the left residual is always less than the unit roundoff, the right residual becomes large as n increases. These matrices are very ill conditioned (singular to working precision for $n \geq 20$), yet it is still reasonable to expect a small residual, and we will prove in §13.3.2 that the left residual must be small, independent of the condition number.

In most of this chapter we are not concerned with the precise values of constants (§13.4 is the exception); thus c_n denotes a constant of order n. To simplify the presentation we introduce a special notation. Let $A_i A_i \in \mathbb{R}^{m_i \times n_i}$, $i = 1{:}k$, be matrices such that the product $A_1 A_2 \ . \ . \ . \ A_k$ is defined and let

$$p = \sum_{i=1}^{k-1} n_i.$$

Then $D(A_1,\ A_2,\ .\ .\ .\ .\ ,\ A_k) \in \mathbb{R}^{m_1 \times n_k}$ denotes a matrix bounded according to

$$|\Delta(A_1, A_2, \ldots, A_k)| \leq c_p u |A_1||A_2| \ldots |A_k|.$$

This notation is chosen so that if $\widehat{C} = fl(A_1A_2\dots A_k)$, with the product evaluated in any order, then

$$\widehat{C} = A_1A_2\dots A_k + \Delta(A_1, A_2, \dots, A_k).$$

13.2. Inverting a Triangular Matrix

We consider the inversion of a lower triangular matrix $L \in \mathbb{R}^{n\times n}$, treating unblocked and blocked methods separately. We do not make a distinction between partitioned and block methods in this section. All the results in this and the next section are from Du Croz and Higham [322, 1992].

13.2.1. Unblocked Methods

We focus our attention on two "*j*" methods that compute L^{-1} a column at a time. Analogous "*i*" and "*k*" methods exist, which compute L^{-1} row-wise or use outer products, respectively, and we comment on them at the end of the section.

The first method computes each column of $X = L^{-1}$ independently, using forward substitution. We write it as follows, to facilitate comparison with the second method.

```
Method 1.
for $j = 1:n$
    $x_{jj} = l_{jj}^{-1}$
    $X(j+1:n, j) = -x_{jj}L(j+1:n, j)$
    Solve $L(j+1:n, j+1:n)X(j+1:n, j) = X(j+1:n, j)$,
    by forward substitution.
end
```

In BLAS terminology, this method is dominated by n calls to the level-2 BLAS routine `xTRSV` (Triangular SolVe).

The second method computes the columns in the reverse order. On the jth step it multiplies by the previously computed inverse $L(j+1:n, j+1:n)^{-1}$ instead of solving a system with coefficient matrix $L(j+1:n, j+1:n)$.

```
Method 2.
for $j = n:-1:1$
    $x_{jj} = l_{jj}^{-1}$
    $X(j+1:n, j) = X(j+1:n, j+1:n)L(j+1:n, j)$
    $X(j+1:n, j) = -x_{jj}X(j+1:n, j)$
end
```

Method 2 uses n calls to the level-2 BLAS routine xTRMV (Triangular Matrix times Vector). On most high-performance machines xTRMV can be implemented to run faster than xTRSV, so Method 2 is generally preferable to Method 1 from the point of view of efficiency (see the performance figures at the end of §13.2.2). We now compare the stability of the two methods.

Theorem 8.5 shows that the *j*th column of the computed $\widehat{X}$ from Method 1 satisfies

$$(L + \Delta L_j)\widehat{x}_j = e_j, \quad |\Delta L_j| \le c_n u|L|.$$

It follows that we have the componentwise residual bound

$$|L\widehat{X} - I| \le c_n u|L||\widehat{X}| \tag{13.4}$$

and the componentwise forward error bound

$$|\widehat{X} - L^{-1}| \le c_n u|L^{-1}||L||\widehat{X}|. \tag{13.5}$$

Since $\widehat{X} = L^{-1} + O(u)$, (13.5) can be written as

$$|\widehat{X} - L^{-1}| \le c_n u|L^{-1}||L||L^{-1}| + O(u^2), \tag{13.6}$$

which is invariant under row and column scaling of L. If we take norms we obtain normwise relative error bounds that are either row or column scaling independent: from (13.6) we have

$$\frac{\|\widehat{X} - L^{-1}\|_\infty}{\|L^{-1}\|_\infty} \le c_n u \operatorname{cond}(L^{-1}) + O(u^2), \tag{13.7}$$

and the same bound holds with cond(L^{-1}) replaced by cond(L).

Notice that (13.4) is a bound for the *right residual*, $LX - I$. This is because Method 1 is derived by solving $LX = I$. Conversely, Method 2 can be derived by solving $XL = I$, which suggests that we should look for a bound on the left residual for this method.

Lemma 13.1. *The computed inverse* $\widehat{X}$ *from Method 2 satisfies*

(13.8)

Proof. The proof is by induction on n, the case $n = 1$ being trivial. Assume the result is true for $n - 1$ and write

$$L = \begin{bmatrix} \alpha & 0 \\ y & M \end{bmatrix}, \qquad X = L^{-1} = \begin{bmatrix} \beta & 0 \\ z & N \end{bmatrix},$$

where $\alpha, \beta \in \mathbb{R}$, $y, z \in \mathbb{R}^{n-1}$ and $M, N \in \mathbb{R}^{(n-1)\times(n-1)}$. Method 2 computes the first column of X by solving $XL = I$ according to

$$b = a^{-1}, \qquad z = -bNy.$$

In floating point arithmetic we obtain

$$\begin{aligned}\widehat{\beta} &= \alpha^{-1}(1+\delta), \qquad |\delta| \le u,\\ \widehat{z} &= -\widehat{\beta}\widehat{N}y + \Delta(\widehat{\beta},\widehat{N},y).\end{aligned}$$

Thus

$$\begin{aligned}\widehat{\beta}\alpha &= 1+\delta,\\ \widehat{z}\alpha + \widehat{N}y &= -\delta\widehat{N}y + \alpha\Delta(\widehat{\beta},\widehat{N},y).\end{aligned}$$

This may be written as

$$\begin{aligned}|\widehat{X}L - I|(1\colon n,1) &\le \begin{bmatrix} u \\ u|\widehat{N}||y| + c_n u(1+u)|\widehat{N}||y| \end{bmatrix}\\ &\le c_n' u\big(|\widehat{X}||L|\big)(1\colon n,1).\end{aligned}$$

By assumption, the corresponding inequality holds for the $(2\colon n,\ 2\colon n)$ submatrices and so the result is proved. □

Lemma 13.1 shows that Method 2 has a left residual analogue of the right residual bound (13.4) for Method 1. Since there is, in general, no reason to choose between a small right residual and a small left residual, our conclusion is that Methods 1 and 2 have equally good numerical stability properties.

More generally, it can be shown that all three i, j, and k inversion variants that can be derived from the equations $LX = I$ produce identical rounding errors under suitable implementations, and all satisfy the same right residual bound; likewise, the three variants corresponding to the equation $XL = I$ all satisfy the same left residual bound. The LINPACK routine `xTRDI` uses a k variant derived from $XL = I$; the LINPACK routines `xGEDI` and `xPODI` contain analogous code for inverting an upper triangular matrix (but the LINPACK Users' Guide [307, 1979, Chaps. 1 and 3] describes a different variant from the one used in the code).

13.2.2. Block Methods

Let the lower triangular matrix $L \in \mathbb{R}^{n\times n}$ be partitioned in block form as

$$L = \begin{bmatrix} L_{11} & & & \\ L_{21} & L_{22} & & \\ \vdots & & \ddots & \\ L_{N1} & \dots & \dots & L_{NN} \end{bmatrix}, \tag{13.9}$$

where we place no restrictions on the block sizes, other than to require the diagonal blocks to be square. The most natural block generalizations of Methods 1 and 2 are as follows. Here, we use the notation $L_{p:q,r:s}$ to denote the submatrix comprising the intersection of block rows p to q and block columns r to s of L.

Method 1B.

```
for $j = 1:N$
    $X_{jj} = L_{jj}^{-1}$ (by Method 1)
    $X_{j+1:N,j} = -L_{j+1:N,j}X_{jj}$
    Solve $L_{j+1:N,j+1:N}X_{j+1:N,j} = X_{j+1:N,j}$,
    by forward substitution
end
```

Method 2B.

```
for $j = N:-1:1$
    $X_{jj} = L_{jj}^{-1}$ (by Method 2)
    $X_{j+1:N,j} = X_{j+1:N,j+1:N}L_{j+1:N,j}$
    $X_{j+1:N,j} = -X_{j+1:N,j}X_{jj}$
end
```

One can argue that Method 1B carries out the same arithmetic operations as Method 1, although possibly in a different order, and that it therefore satisfies the same error bound (13.4). For completeness, we give a direct proof.

Lemma 13.2. *The computed inverse* $\widehat{X}$ *from Method* 1B *satisfies*

$$|L\widehat{X} - I| \le c_n u|L||\widehat{X}|. \tag{13.10}$$

Proof. Equating block columns in (13.10), we obtain the N independent inequalities

$$|L\widehat{X}_{1:N,j} - I_{1:N,j}| \le c_n u|L||\widehat{X}_{1:N,j}|, \quad j = 1:N. \tag{13.11}$$

It suffices to verify the inequality with $j = 1$. Write

$$L = \begin{bmatrix} L_{11} & \\ L_{21} & L_{22} \end{bmatrix}, \qquad X = \begin{bmatrix} X_{11} & \\ X_{21} & X_{22} \end{bmatrix},$$

where L_{11}, $X_{11} \in \mathbb{R}^{r\times r}$ and L_{11} is the (1, 1) block in the partitioning of (13.9). X_{11} is computed by Method 1 and so, from (13.4),

$$|L_{11}\widehat{X}_{11} - I| \le c_r u|L_{11}||\widehat{X}_{11}| = c_r u\big(|L||\widehat{X}|\big)_{11}. \tag{13.12}$$

X_{21} is computed by forming $T = -L_{21}X_{11}$ and solving $L_{22}X_{21} = T$. The computed X_{21} satisfies

$$L_{22}\widehat{X}_{21} + \Delta(L_{22}, \widehat{X}_{21}) = -L_{21}\widehat{X}_{11} + \Delta(L_{21}, \widehat{X}_{11}).$$

Hence

$$\begin{aligned}|L_{21}\widehat{X}_{11} + L_{22}\widehat{X}_{21}| &\le c_n u\bigl(|L_{21}||\widehat{X}_{11}| + |L_{22}||\widehat{X}_{21}|\bigr)\\ &= c_n u\bigl(|L||\widehat{X}|\bigr)_{21}.\end{aligned} \tag{13.13}$$

Together, inequalities (13.12) and (13.13) are equivalent to (13.11) with $j = 1$, as required. □

We can attempt a similar analysis for Method 2B. With the same notation as above, X_{21} is computed as $X_{21} = -X_{22}L_{21}X_{11}$. Thus

$$\widehat{X}_{21} = -\widehat{X}_{22}L_{21}\widehat{X}_{11} + \Delta(\widehat{X}_{22}, L_{21}, \widehat{X}_{11}). \tag{13.14}$$

To bound the left residual we have to postmultiply by L_{11} and use the fact that X_{11} is computed by Method 2:

$$\widehat{X}_{21}L_{11} + \widehat{X}_{22}L_{21}(I + \Delta(\widehat{X}_{11}, L_{11})) = \Delta(\widehat{X}_{22}, L_{21}, \widehat{X}_{11})L_{11}.$$

This leads to a bound of the form

$$|\widehat{X}_{21}L_{11} + \widehat{X}_{22}L_{21}| \le c_n u|\widehat{X}_{22}||L_{21}||\widehat{X}_{11}||L_{11}|,$$

which would be of the desired form in (13.8) were it not for the factor $|\widehat{X}_{11}||L_{11}|$. This analysis suggests that the left residual is not guaranteed to be small. Numerical experiments confirm that the left and right residuals can be large simultaneously for Method 2B, although examples are quite hard to find [322, 1992]; therefore the method must be regarded as unstable when the block size exceeds 1.

The reason for the instability is that there are *two* plausible block generalizations of Method 2 and we have chosen an unstable one that does not carry out the same arithmetic operations as Method 2. If we perform a solve with L_{jj} instead of multiplying by X_{jj} we obtain the second variation, which is used by LAPACK's `xTRTRI`:

```
Method 2C.
for $j = N:-1:1$
    $X_{jj} = L_{jj}^{-1}$ (by Method 2)
    $X_{j+1:N,j} = X_{j+1:N,j+1:N}L_{j+1:N,j}$
    Solve $X_{j+1:N,j}L_{jj} = -X_{j+1:N,j}$ by back substitution.
end
```

For this method, the analogue of (13.14) is

$$\widehat{X}_{21}L_{11} + \Delta(\widehat{X}_{21}, L_{11}) = -\widehat{X}_{22}L_{21} + \Delta(\widehat{X}_{22}, L_{21})$$

which yields

$$|\widehat{X}_{21}L_{11} + \widehat{X}_{22}L_{21}| \le c_n u\bigl(|\widehat{X}_{21}||L_{11}| + |\widehat{X}_{22}||L_{21}|\bigr).$$

Hence Method 2C enjoys a very satisfactory residual bound.

Table 13.2. *Mflop rates for inverting a triangular matrix on a Cray 2.*

		$n = 128$	$n = 256$	$n = 512$	$n = 1024$
Unblocked:	Method 1	95	162	231	283
	Method 2	114	211	289	330
	k variant	114	157	178	191
Blocked:	Method 1B	125	246	348	405
(block size 64)	Method 2C	129	269	378	428
	k variant	148	263	344	383

Lemma 13.3. *The computed inverse* $\widehat{X}$ *from Method* 2C *satisfies*

$$|\widehat{X}L - I| \le c_n u|\widehat{X}||L|. \qquad \square$$

In summary, block versions of Methods 1 and 2 are available that have the same residual bounds as the point methods. However, in general, there is no guarantee that stability properties remain unchanged when we convert a point method to block form, as shown by Method 2B.

In Table 13.2 we present some performance figures for inversion of a lower triangular matrix on a Cray 2. These clearly illustrate the possible gains in efficiency from using block methods, and also the advantage of Method 2 over Method 1. For comparison, the performance of a k variant is also shown (both k variants run at the same rate). The performance characteristics of the i variants are similar to those of the j variants, except that since they are row oriented rather than column oriented, they are liable to be slowed down by memory-bank conflicts, page thrashing, or cache missing.

13.3. Inverting a Full Matrix by LU Factorization

Next, we consider four methods for inverting a full matrix $A \in \mathbb{R}^{n\times n}$, given an LU factorization computed by GEPP. We assume, without loss of generality, that there are no row interchanges. We write the *computed* LU factors as L and U. Recall that $A + DA = LU$, with $|DA| \le c_n u|L||U|$ (Theorem 9.3).

13.3.1. Method A

Perhaps the most frequently described method for computing $X = A^{-1}$ is the following one.

Method A.

```
for $j = 1:n$
```

```
    Solve $Ax_J = e_j$
end
```

Compared with the methods to be described below, Method A has the disadvantages of requiring more temporary storage and of not having a convenient partitioned version. However, it is simple to analyse. From Theorem 9.4 we have

$$(A + \Delta A_j)\widehat{x}_j = e_j, \qquad |\Delta A_j| \le c_n' u|L||U|, \tag{13.15}$$

and so

$$|A\widehat{X} - I| \le c_n' u|L||U||\widehat{X}|. \tag{13.16}$$

This bound departs from the form (13. 1) only in that $|A|$ is replaced by its upper bound $|L||U| + O(u)$. The forward error bound corresponding to (13.16) is

$$|\widehat{X} - A^{-1}| \le c_n' u|A^{-1}||L||U||\widehat{X}|. \tag{13.17}$$

Note that (13.15) says that $\widehat{x}_j$ is the jth column of the inverse of a matrix close to A, but it is a different perturbation DA_j for each column. It is not true that $\widehat{X}$ itself is the inverse of a matrix close to A, unless A is well conditioned.

13.3.2. Method B

Next, we consider the method used by LINPACK'S `xGEDI`, LAPACK'S `xGETRI`, and MATLAB'S **INV** function.

Method B.
Compute U^{-1} and then solve for X the equation $XL = U^{-1}$.

To analyse this method we will assume that U^{-1} is computed by an analogue of Method 2 or 2C for upper triangular matrices that obtains the columns of U^{-1} in the order 1 to n. Then the computed inverse $Xu \approx U^{-1}$ will satisfy the residual bound

$$|X_U U - I| \le c_n u|X_U||U|.$$

We also assume that the triangular solve from the right with L is done by back substitution. The computed X therefore satisfies $XL = Xu + D(X, L)$ and so

$$\widehat{X}(A + \Delta A) = \widehat{X}LU = X_U U + \Delta(\widehat{X}, L)U.$$

This leads to the residual bound

$$\begin{aligned} |\widehat{X}A - I| &\le c_n u\big(|U^{-1}||U| + 2|\widehat{X}||L||U|\big) \\ &\le c_n' u|\widehat{X}||L||U|, \end{aligned} \tag{13.18}$$

which is the left residual analogue of (13.16). From (13.18) we obtain the forward error bound

$$|\widehat{X} - A^{-1}| \le c_n' u |\widehat{X}||L||U||A^{-1}|.$$

Note that Methods A and B are equivalent, in the sense that Method A solves for X the equation $LUX = I$ while Method B solves $XLU = I$. Thus the two methods carry out analogous operations but in different orders. It follows that the methods must satisfy analogous residual bounds, and so (13.18) can be deduced from (13.16).

We mention in passing that the LINPACK manual states that for Method B a bound holds of the form $\|AX - I\| \le d_n u \|A\| \, \|X\|$ [307, 1979, p. 1.20]. This is incorrect, although counterexamples are rare; it is the left residual that is bounded this way, as follows from (13.18).

13.3.3. Method C

The next method that we consider is from Du Croz and Higham [322, 1992]. It solves the equation $UXL = I$, computing X a partial row and column at a time. To derive the method partition

$$X = \begin{bmatrix} x_{11} & x_{12}^T \\ x_{21} & X_{22} \end{bmatrix}, \qquad L = \begin{bmatrix} 1 & 0 \\ l_{21} & L_{22} \end{bmatrix}, \qquad U = \begin{bmatrix} u_{11} & u_{12}^T \\ 0 & U_{22} \end{bmatrix},$$

where the (1, 1) blocks are scalars, and assume that the trailing submatrix X_{22} is already known. Then the rest of X is computed according to

$$\begin{aligned} x_{21} &= -X_{22} l_{21}, \\ x_{12}^T &= -u_{12}^T X_{22} / u_{11}, \\ x_{11} &= 1/u_{11} - x_{12}^T l_{21}. \end{aligned}$$

The method can also be derived by forming the product $X = U^{-1} \times L^{-1}$ using the representation of L and U as a product of elementary matrices (and diagonal matrices in the case of U). In detail the method is as follows.

Method C.

```
for k = n:-1:1
    $X(k+1:n, k) = -X(k+1:n, k+1:n)L(k+1:n, k)$
    $X(k, k+1:n) = -U(k, k+1:n)X(k+1:n, k+1:n)/u_{kk}$
    $x_{kk} = 1/u_{kk} - X(k, k+1:n)L(k+1:n, k)$
end
```

The method can be implemented so that X overwrites L and U, with the aid of a work vector of length n (or a work array to hold a block row or column in the partitioned case). Because most of the work is performed by matrix–vector (or matrix-matrix) multiplication, Method C is likely to be the fastest of those considered in this section on many machines. (Some performance figures are given at the end of the section.)

A straightforward error analysis of Method C shows that the computed $\widehat{X}$ satisfies

$$|U\widehat{X}L - I| \le c_n u |U||\widehat{X}||L|. \tag{13.19}$$

We will refer to $U\widehat{X}L - I$ as a "mixed residual". From (13.19) we can obtain bounds on the left and right residual that are weaker than those in (13.18) and (13.16) by a factor $|U^{-1}||U|$ on the left or $|L||L^{-1}|$ on the right, respectively. We also obtain from (13.19) the forward error bound

$$|\widehat{X} - A^{-1}| \le c_n u |U^{-1}||U||\widehat{X}||L||L^{-1}|,$$

which is (13.17) with $|A^{-1}|$ replaced by its upper bound $|U^{-1}||L^{-1}| + O(u)$ and the factors reordered.

The LINPACK routine `xSIDI` uses a special case of Method C in conjunction with the diagonal pivoting method to invert a symmetric indefinite matrix; see Du Croz and Higham [322, 1992] for details.

13.3.4. Method D

The next method is based on another natural way to form A^{-1} and is used by LAPACK'S `xPOTRI`, which inverts a symmetric positive definite matrix.

Method D.
Compute L^{-1} and U^{-1} and then form $A^{-1} = U^{-1} \times L^{-1}$.

The advantage of this method is that no extra workspace is needed; U^{-1} and L^{-1} can overwrite U and L, and can then be overwritten by their product. However, Method D is significantly slower on some machines than Methods B or C, because it uses a smaller average vector length for vector operations.

To analyse Method D we will assume initially that L^{-1} is computed by Method 2 (or Method 2C) and, as for Method B above, that U^{-1} is computed by an analogue of Method 2 or 2C for upper triangular matrices. We have

$$\widehat{X} = X_U X_L + \Delta(X_U, X_L). \tag{13.20}$$

Since $A = LU - DA$,

$$\widehat{X}A = X_U X_L LU - X_U X_L \Delta A + \Delta(X_U, X_L)A. \tag{13.21}$$

Rewriting the first term of the right-hand side using $X_L L = I + D(X_L, L)$, and similarly for U, we obtain

$$\widehat{X}A - I = \Delta(X_U, U) + X_U \Delta(X_L, L)U - X_U X_L \Delta A + \Delta(X_U, X_L)A, \tag{13.22}$$

and so

$$
\begin{aligned}
|\widehat{X}A - I| &\le c_n' u\big(|U^{-1}||U| + 2|U^{-1}||L^{-1}||L||U| + |U^{-1}||L^{-1}||A|\big)\\
&\le c_n'' u|U^{-1}||L^{-1}||L||U|. \qquad (13.23)
\end{aligned}
$$

This bound is weaker than (13.18), since $|\widehat{X}| \le |U^{-1}||L^{-1}| + O(u)$. Note, however, that the term $D(X_U,X_L)A$ in the residual (13.22) is an unavoidable consequence of forming $X_U X_L$, and so the bound (13.23) is essentially the best possible.

The analysis above assumes that X_L and X_U both have small left residuals. If they both have small right residuals, as when they are computed using Method 1, then it is easy to see that a bound analogous to (13.23) holds for the right residual $A\widehat{X} - I$. If X_L has a small left residual and X_U has a small right residual (or vice versa) then it does not seem possible to derive a bound of the form (13.23). However, we have

$$|X_L L - I| = |L^{-1}(L X_L - I)L| \le |L^{-1}||LX_L - I||L|, \qquad (13.24)$$

and since L is unit lower triangular with $|l_{ij}| \le 1$, we have $|(L^{-1})_{ij}| \le 2^{n-1}$, which places a bound on how much the left and right residuals of X_L can differ. Furthermore, since the matrices L from GEPP tend to be well conditioned $(\kappa_\infty(L) \ll n2^{n-1})$ and since our numerical experience is that large residuals tend to occur only for ill-conditioned matrices, we would expect the left and right residuals of XL almost always to be of similar size. We conclude that even in the "conflicting residuals" case, Method D will, in practice, usually satisfy (13.23) or its right residual counterpart, according to whether X_U has a small left or right residual respectively. Similar comments apply to Method B when U^{-1} is computed by a method yielding a small right residual.

These considerations are particularly pertinent when we consider Method D specialized to symmetric positive definite matrices and the Cholesky factorization $A = R^T R$. Now A^{-1} is obtained by computing $X_R = R^{-1}$ and then forming $A^{-1} = X_R X_R^T$; this is the method used in the LINPACK routine xPODI [307, 1979, Chap. 3]. If X_R has a small right residual then X_R^T has a small left residual, so in this application we naturally encounter conflicting residuals. Fortunately, the symmetry and definiteness of the problem help us to obtain a satisfactory residual bound. The analysis parallels the derivation of (13.23), so it suffices to show how to treat the term $X_R X_R^T R^T R$ (cf. (13.21)), where R now denotes the computed Cholesky factor. Assuming $RX_R = I + D(R, X_R)$, and using (13.24) with L replaced by R, we have

$$
\begin{aligned}
X_R X_R^T R^T R &= X_R\big(I + \Delta(R, X_R)^T\big)R\\
&= I + F + X_R \Delta(R, X_R)^T R, \qquad |F| \le |R^{-1}||\Delta(R, X_R)||R|,\\
&= I + G,
\end{aligned}
$$

Table 13.3. *Mflop rates for inverting a full matrix on a Cray 2.*

		$n = 64$	$n = 128$	$n = 256$	$n = 512$
Unblocked:	Method B	118	229	310	347
	Method C	125	235	314	351
	Method D	70	166	267	329
Blocked:	Method B	142	259	353	406
(block size 64)	Method C	144	264	363	415
	Method D	70	178	306	390

and

$$|G| \le c_n u(|R^{-1}||R||R^{-1}||R| + |R^{-1}||R^{-T}||R^T||R|).$$

From the inequality $\| |B| \|_2 \le \sqrt{n}\|B\|_2$ for $B \in \mathbb{R}^{n\times n}$, together with $\|A\|_2 = \|R\|_2^2 + O(u)$, it follows that

$$\|G\|_2 \le 2n^2 c_n u \|A\|_2 \|A^{-1}\|_2,$$

and thus overall we have a bound of the form

$$\|\widehat{X}A - I\|_2 \le d_n u \|A\|_2 \|\widehat{X}\|_2.$$

Since $\widehat{X}$ and A are symmetric the same bound holds for the right residual.

13.3.5. Summary

In terms of the error bounds, there is little to choose between Methods A, B, C, and D. Numerical results reported in [322, 1992] show good agreement with the bounds. Therefore the choice of method can be based on other criteria, such as performance and the use of working storage. Table 13.3 gives some performance figures for a Cray 2, covering both blocked (partitioned) and unblocked forms of Methods B, C, and D.

On a historical note, Tables 13.4 and 13.5 give timings for matrix inversion on some early computing devices; times for two modern machines are given for comparison. The inversion methods used for the timings on the early computers in Table 13.4 are not known, but are probably methods from this section or the next.

13.4. Gauss–Jordan Elimination

Whereas Gaussian elimination (GE) reduces a matrix to triangular form by elementary operations, Gauss–Jordan elimination (GJE) reduces it all the way

Table 13.4. *Times (minutes and seconds) for inverting an n × n matrix. Source for DEUCE, Pegasus, and Mark* 1 *timings:* [181, 1981].

n	DEUCE (English Electric) 1955	Pegasus (Ferranti) 1956	Manchester Mark 1 1951	HP 48G Calculator 1993	Sun SPARC-station ELC 1991
8	20s	26s	–	4s	.004s
16	58s	2m 37s	–	18s	.01s
24	3m 36s	7m 57s	8m	48s	.02s
32	7m 44s	17m 52s	16m	–	.04s

Table 13.5. *Additional timings for inverting an n × n matrix.*

Machine	Year	n	Time	Reference
Aiken Relay Calculator	1948	38	59½ hours	[764, 1948]
IBM 602 Calculating Punch	1949	10	8 hours	[1053, 1949]
SEAC (National Bureau of Standards)	1954	49	3 hours	[1004, 1954]
Datatron	1957	80’	2½ hours	[753, 1957]
IBM 704	1957	115*	19m 30s	[320, 1957]

---

[a] Block tridiagonal matrix, using an inversion method designed for such matrices.
asymmetric positive definite matrix.

to diagonal form. GJE is usually presented as a method for matrix inversion, but it can also be regarded as a method for solving linear equations. We will take the method in its latter form, since it simplifies the error analysis. Error bounds for matrix inversion are obtained by taking unit vectors for the right-hand sides.

At the kth stage of GJE, all off-diagonal elements in the kth column are eliminated, instead of just those below the diagonal, as in GE. Since the elements in the lower triangle (including the pivots on the diagonal) are identical to those that occur in GE, Theorem 9.1 tells us that GJE succeeds if all the leading principal submatrices of A are nonsingular. With no form of pivoting GJE is unstable in general, for the same reasons that GE is unstable. Partial and complete pivoting are easily incorporated.

Algorithm 13.4 (Gauss–Jordan elimination). This algorithm solves the linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is nonsingular, by GJE with partial pivoting.

```
for $k = 1:n$
    Find $r$ such that $|a_{rk}| = \max_{i\ge k} |a_{ik}|$.
    $A(k, k:n) \leftrightarrow A(r, k:n)$, $b(k) \leftrightarrow b(r)$ % Swap rows $k$ and $r$.
    $row\_ind = [1:k-1,\ k+1:n]$ % Row indices of elements to eliminate.
    $m = A(row\_ind,k)/A(k,k)$ % Multipliers.
    $A(row\_ind,k:n) = A(row\_ind,k:n) - m*A(k,k:n)$
    $b(row\_ind) = b(row\_ind) - m*b(k)$
end
$x_i = b_i/a_{ii}$, $i = 1:n$
```

Cost: n^3 flops.

The numerical stability properties of GJE are rather subtle and error analysis is trickier than for GE. An observation that simplifies the analysis is that we can consider the algorithm as comprising two stages. The first stage is identical to GE and forms $M_{n-1}M_{n-2}\dots M_1A = U$, where U is upper triangular. The second stage reduces U to diagonal form by elementary operations:

$$N_nN_{n-1}\dots N_2U = D, \qquad N_k = I - n_ke_k^T, \quad e_i^Tn_k = 0,\ i \ge k.$$

The solution x is formed as $x = D^{-1}N_n\dots N_2y$, where $y = M_{n-1}\dots M_2b$. The rounding errors in the first stage are precisely the same as those in GE, so it suffices to consider the second stage. We will assume, for simplicity, that there are no row interchanges. As with GE, this is equivalent to assuming that all row interchanges are done at the start of the algorithm.

Define $U_{k+1} = N_k\dots N_2U$ (so $U_2 = U$) and note that N_k and U_k have the forms

$$U_k = \begin{bmatrix} D_{k-1} & [v_k \ \ W_k] \\ 0 & Z_k \end{bmatrix}, \quad D_{k-1} \in \mathbb{R}^{(k-1)\times(k-1)}, \text{ diagonal,}$$
$$N_k = \begin{bmatrix} I_{k-1} & [n_k \ \ 0] \\ 0 & I_{n-k+1} \end{bmatrix},$$

where $n_k = [-u_{1k}/u_{kk}, \dots, -u_{k-1,k}/u_{kk}]^T$. The computed matrices obviously satisfy

$$\widehat{U}_{k+1} = \widehat{N}_k\widehat{U}_k + \Delta_k, \tag{13.25a}$$
$$\Delta_k = \begin{bmatrix} 0_{k-1} & \Delta^{(k)} \\ 0 & 0_{n-k+1} \end{bmatrix}, \quad |\Delta_k| \le \gamma_3|\widehat{N}_k||\widehat{U}_k|, \tag{13.25b}$$

(to be precise, $\widehat{N}_k$ is defined as N_k but with $(\widehat{N}_k)_{ik} = -\widehat{u}_{ik}/\widehat{u}_{kk}$). Similarly, with $x_{k+1} = N_k\dots N_2y$, we have

$$\widehat{x}_{k+1} = \widehat{N}_k\widehat{x} + f_k, \quad e_i^Tf_k = 0,\ i \ge k, \quad |f_k| \le \gamma_3|\widehat{N}_k||\widehat{x}_k|. \tag{13.26}$$

Because of the structure of $\widehat{N}_k$, D_k, and f_k, we have the useful property that

$$\widehat{N}_i \Delta_j = \Delta_j, \quad i \ge j, \qquad \widehat{N}_i f_j = f_j, \quad i \ge j.$$

Without loss of generality, we now assume that the final diagonal matrix D is the identity (i.e., the pivots are all 1); this simplifies the analysis a little and has a negligible effect on the final bounds. Thus (13.25) and (13.26) yield

$$I = \widehat{U}_{n+1} = \widehat{N}_n \dots \widehat{N}_2 U + \sum_{k=2}^{n} \Delta_k, \tag{13.27}$$

$$\widehat{x} = \widehat{x}_{n+1} = \widehat{N}_n \dots \widehat{N}_2 y + \sum_{k=2}^{n} f_k. \tag{13.28}$$

Now

$$\begin{aligned} |\Delta_k| &\le \gamma_3 |\widehat{N}_k||\widehat{U}_k| \le \gamma_3 |\widehat{N}_k||\widehat{N}_{k-1}\widehat{U}_{k-1} + \Delta_{k-1}| \\ &\le \gamma_3(1+\gamma_3)|\widehat{N}_k||\widehat{N}_{k-1}|\widehat{U}_{k-1}| \\ &\le \cdots \le \gamma_3(1+\gamma_3)^{k-2}|\widehat{N}_k|\dots|\widehat{N}_2||U|, \end{aligned}$$

and, similarly,

$$|f_k| \le \gamma_3(1+\gamma_3)^{k-2}|\widehat{N}_k|\dots|\widehat{N}_2||y|.$$

But defining $\tilde{n}_k = [n_k^T \ 0]^T \in \mathbb{R}^n$, we have

$$\begin{aligned} |\widehat{N}_k|\dots|\widehat{N}_2| &= I + |\widetilde{n}_k|e_k^T + \cdots + |\widetilde{n}_2|e_2^T \\ &\le |\widehat{N}_n|\dots|\widehat{N}_2| = |\widehat{N}_n \dots \widehat{N}_2| =: |X|, \end{aligned}$$

where $X = U^{-1} + O(u)$ (by (13.27)). Hence

$$\sum_{k=2}^{n} |\Delta_k| \le (n-1)\gamma_3(1+\gamma_3)^{n-2}|X||U|,$$

$$\sum_{k=2}^{n} |f_k| \le (n-1)\gamma_3(1+\gamma_3)^{n-2}|X||y|.$$

Combining (13.27) and (13.28) we have, for the solution of $U_x = y$,

$$\widehat{x} = \Big(I - \sum_{k=2}^{n} \Delta_k\Big)U^{-1}y + \sum_{k=2}^{n} f_k = \Big(I - \sum_{k=2}^{n} \Delta_k\Big)x + \sum_{k=2}^{n} f_k,$$

which gives the componentwise forward error bound

$$|x - \widehat{x}| \le (n-1)\gamma_3(1+\gamma_3)^{n-2}|X|(|U||x| + |y|). \tag{13.29}$$

Table 13.6. *Gauss-Jordan elimination for* $U_x = b$.

n	$h_{U,b}(\widehat{x})$	$\kappa_\infty(U)u$
16	2.0e-14	5.8e-11
32	6.4e-10	7.6e-6
64	1.7e-2	6.6e4

This is an excellent forward error bound: it says that the error in $\widehat{x}$ is no larger than the error we would expect for an approximate solution with a tiny componentwise relative backward error. In other words, the forward error is bounded in the sane way as for substitution. However, we have not, and indeed cannot, show that the method is backward stable. The best we can do is to derive from (13.27) and (13.28) the result

$$(U + \Delta U)\widehat{x} = y + \Delta y, \tag{13.30a}$$

$$|\Delta U| \le (n-1)\gamma_3(1+\gamma_3)^{n-2}|U||X||U|, \tag{13.30b}$$

$$|\Delta y| \le (n-1)\gamma_3(1+\gamma_3)^{n-2}|U||X||y|, \tag{13.30C}$$

using $\widehat{N}_2^{-1}\dots\widehat{N}_n^{-1} = U + O(u)$. These bounds show that $\widehat{x}$ has a normwise backward error bounded approximately by $\kappa_\infty(U)(n-1)\gamma_3$. Hence the backward error can be guaranteed to be small only if U is well conditioned. This agrees with the comments of Peters and Wilkinson [828, 1975] that "the residuals corresponding to the Gauss-Jordan solution are often larger than those corresponding to back-substitution by a factor of order κ."

A numerical example helps to illustrate the results. We take U to be the upper triangular matrix with 1s on the diagonal and – 1s everywhere above the diagonal (U = $U(1)$ from (8.2)). This matrix has condition number $\kappa_\infty(U) = n2^{n-1}$. Table 13.6 reports the normwise relative backward error $\eta_{U,b}(\widehat{x}) = \|U\widehat{x} - b\|_\infty/(\|U\|_\infty\|\widehat{x}\|_\infty + \|b\|_\infty)$ (see (7.2)) for $b = U_x$, where $x = e/3$. Clearly, GJE is backward unstable for these matrices—the backward errors show a dependence on $\kappa_\infty(U)$. However, the relative distance between $\widehat{x}$ and the computed solution from substitution (not shown in the table) is less than $\kappa_\infty(U)u$, which shows that the forward error is bounded by $\kappa_\infty(U)u$, confirming (13.29).

By bringing in the error analysis for the reduction to upper triangular form, we obtain an overall result for GJE.

Theorem 13.5. *Suppose GJE successfully computes an approximate solution* $\widehat{x}$ *to* $Ax = b$, *where* $A \in \mathbb{R}^{n\times n}$ *is nonsingular. Then*

$$|b - A\widehat{x}| \le 8nu|\widehat{L}||\widehat{U}||\widehat{U}^{-1}||\widehat{U}||\widehat{x}| + O(u^2), \tag{13.31}$$

$$|x - \widehat{x}| \le \bigl(2n|A^{-1}||\widehat{L}||\widehat{U}| + 6n(|\widehat{U}^{-1}||\widehat{U}|)^2\bigr)|\widehat{x}| + O(u^2), \tag{13.32}$$

where $A \approx \widehat{L}\widehat{U}$ *is the factorization computed by GE.*

Proof. For the first stage (which is just GE), we have $A + DA_1 = \widehat{L}\widehat{U}$, $|\Delta A_1| \le \gamma_n|\widehat{L}||\widehat{U}|$, and $(\widehat{L} + \Delta L)\widehat{y} = b$, $|\Delta L| \le \gamma_n|L|$, by Theorems 9.3 and 8.5.

Using (13.30), we obtain

$$(\widehat{L} + \Delta L)(\widehat{U} + \Delta U)\widehat{x} = (\widehat{L} + \Delta L)(\widehat{y} + \Delta y) = b + (\widehat{L} + \Delta L)\Delta y,$$

or $A\widehat{x} = b - r$, where

$$r = (\Delta A_1 + \widehat{L}\Delta U + \Delta L\widehat{U} + \Delta L\Delta U)\widehat{x} - (\widehat{L} + \Delta L)\Delta y. \tag{13.33}$$

The bounds (13.31) and (13.32) follow easily on using (13.30). □

Theorem 13.5 shows that the stability of GJE depends not only on the size of $|L||U|$ (as in GE), but also on the condition of U. The term $|\widehat{U}^{-1}||\widehat{U}||\widehat{x}|$ is an upper bound for $|\widehat{x}|$, and if this bound is sharp then the residual bound is very similar to that for LU factorization. Note that for partial pivoting we have $\kappa_\infty(\widehat{U}) \le n2^{n-1}\kappa_\infty(A) + O(u)$.

The bounds in Theorem 13.5 have the pleasing property that they are invariant under row or column scaling of A, though of course if we are using partial pivoting then row scaling can change the pivots and alter the bound.

As mentioned earlier, to obtain bounds for matrix inversion we simply take b to be each of the unit vectors in turn. For example, the residual bound becomes

$$|A\widehat{X} - I| \le 8nu|\widehat{L}||\widehat{U}||\widehat{U}^{-1}||\widehat{U}||\widehat{X}| + O(u^2).$$

For the special case of symmetric positive definite matrices, an informative normwise result follows from Theorem 13.5. We make the natural assumption that symmetry is exploited in the elimination.

Corollary 13.6. *Suppose GJE successfully computes an approximate solution* $\widehat{x}$ *to* $Ax = b$, *where* $A \in \mathbb{R}^{n\times n}$ *is symmetric positive definite. Then*

$$\|b - A\widehat{x}\|_2 \le 8n^3u\kappa_2(A)^{1/2}\|A\|_2\|x\|_2 + O(u^2),$$
$$\frac{\|x - \widehat{x}\|_2}{\|x\|_2} \le 8n^3u\kappa_2(A) + O(u^2),$$

where $A \approx \widehat{L}\widehat{U}$ *is the factorization computed by symmetric GE.*

Proof. By Theorem 9.3 we have $A + DA = \widehat{L}\widehat{U}$, where DA is symmetric and satisfies $|DA| \le \gamma_n|\widehat{L}||\widehat{U}|$. Defining $D = \mathrm{diag}(\widehat{U})^{1/2}$ we have, by symmetry, $A + DA = \widehat{L}D \cdot D^{-1}\widehat{U} \equiv .R^TR$. Hence

$$\begin{aligned} \| |\widehat{U}^{-1}||\widehat{U}| \|_2 &= \| |R^{-1}D^{-1}||DR| \|_2 \\ &= \| |R^{-1}||R| \|_2 \\ &\le n\kappa_2(R) = n\kappa_2(A+\Delta A)^{1/2} \\ &= n\kappa_2(A)^{1/2} + O(u). \end{aligned}$$

Furthermore, it is staightforward to show that $\| |\widehat{L}||\widehat{U}| \|_2 = \| |R^T||R| \|_2 \le n(1-\gamma_n)^{-1}\|A\|_2$. The required bounds follow.

Corollary 13.6 shows that GJE is forward stable for symmetric positive definite matrices, but it bounds the backward error only by a multiple of $\kappa_2(A)^{1/2}$. Numerical experiments show that the backward error is usually much less than $\kappa_2(A)^{1/2}u$, but (very ill-conditioned) matrices can certainly be found for which the backward error is many order of magnitude larger than u. Hence GJE is not backward stable even for symmetric positive definite matrices.

13.5. The Determinant

> *It may be too optimistic to hope that determinants will*
> *fade out of the mathematical picture in a generation;*
> *their notation alone is a thing of beauty*
> *to those who can appreciate that sort of beauty.*
>
> — E. T. BELL, *Review of "Contributions to the History of Determinants, 1900–1920", by Sir Thomas Muir (1931)*

Like the matrix inverse, the determinant is a quantity that rarely needs to be computed. The common fallacy that the determinant is a measure of ill conditioning is displayed by the observation that if $Q \in \mathbb{R}^{n\times n}$ is orthogonal then $\det(\alpha Q) = \alpha^n \det(Q) = \pm\alpha^n$, which can be made arbitrarily small or large despite the fact that αQ is perfectly conditioned. Of course, we could normalize the matrix before taking its determinant and define, for example,

$$\psi(A) := \frac{1}{\det(D^{-1}A)} = \frac{\det(D)}{\det(A)}, \qquad D = \mathrm{diag}(\|A(i,:)\|_2),$$

where $D^{-1}A$ has rows of unit 2-norm. This function ψ is called the Hadamard condition number by Birkhoff [99, 1975], because Hadamard's determinantal inequality (see Problem 13.11) implies that $\psi(A) \ge 1$, with equality if and only if A has mutually orthogonal rows. Unless A is already row equilibrated (see Problem 13.13), $\psi(A)$ does not relate to the conditioning of linear systems in any straightforward way.

As good a way as any to compute the determinant of a general matrix $A \in \mathbb{R}^{n\times n}$ is via GEPP. If $PA = LU$ then

$$\det(A) = \det(P)^{-1} \det(U) = (-1)^r u_{11} \ . \ . \ .u_{nn}, \qquad (13.34)$$

where r is the number of row interchanges during the elimination. If we use (13.34) then, barring underflow and overflow, the computed determinant $\widehat{d} = fl(\det(A))$ satisfies

$$\widehat{d} = \det(\widehat{U})(1+\theta_n),$$

where $|q_n| \le \gamma_n$, so we have a tiny relative perturbation of the exact determinant of a perturbed matrix $A + DA$, where DA is bounded in Theorem 9.3. In other words, we have almost the right determinant for a slightly perturbed matrix (assuming the growth factor is small).

However, underflow and overflow in computing the determinant are quite likely, and the determinant itself may not be a representable number. One possibility is to compute $\log |\det(A) = \sum_{i=1}^{n} \log |u_{ii}|$; as pointed out by Forsythe and Moler [396, 1967, p. 55], however, the computed sum may be inaccurate due to cancellation. Another approach, used in LINPACK, is to compute and represent $\det(A)$ in the form $y \times 10^e$, where $1 \le |y| < 10$ and e is an integer exponent.

13.5.1. Hyman's Method

In §1.16 we analysed the evaluation of the determinant of a Hessenberg matrix H by GE. Another method for computing $\det(H)$ is Hyman's method [594, 1957], which is superficially similar to GE but algorithmically different. Hyman's method has an easy derivation in terms of LU factorization that leads to a very short error analysis. Let $H \in \mathbb{R}^{n\times n}$ bean unreduced upper Hessenberg matrix ($h_{i+1}, i \ne 0$ for all i) and write

$$H = \begin{bmatrix} h^T & \eta \\ T & y \end{bmatrix}, \qquad H_1 = \begin{bmatrix} T & y \\ h^T & \eta \end{bmatrix}, \qquad h, y \in \mathbb{R}^{n-1},\ \eta \in \mathbb{R}.$$

The matrix H_1 is H with the first row cyclically permuted to the bottom, so $\det(H_1) = (-1)^{n-1} \det(H)$. Since T is a nonsingular upper triangular matrix, we have the LU factorization

$$H_1 = \begin{bmatrix} I & 0 \\ h^T T^{-1} & 1 \end{bmatrix} \begin{bmatrix} T & y \\ 0 & \eta - h^T T^{-1} y \end{bmatrix} \equiv LU, \qquad (13.35)$$

from which we obtain $\det(H_1) = \det(T)(h - h^T T^{-1} y)$. Therefore

$$\det(H) = (-1)^{n-1} \det(T)(h - h^T T^{-1} y). \qquad (13.36)$$

Hyman's method consists of evaluating (13.36) in the natural way: by solving the triangular system $Tx = y$ by back substitution, then forming $h - h^T x$ and its product with $\det(T) = h_{21}h_{32} \ . \ . \ . \ h_{n,n-1}$.

For the error analysis we assume that no underflows or overflows occur. From the backward error analysis for substitution (Theorem 8.5) and for an inner product ((3.4)) we have immediately, on defining $m := h - h^T T^{-1} y$,

$$\widehat{\mu} = \big(\eta - (h + \Delta h)^T (T + \Delta T)^{-1} y\big)(1 + \delta),$$

where

$$|\Delta h| \le \gamma_{n-1} |h|, \quad |\Delta T| \le \gamma_{n-1} |T|, \quad |\delta| \le u.$$

Since $fl(\det(T)) = \det(T)(1 + \delta_1) . \ . \ .(1 + \delta_{n-1})$, $|\delta_i| \le u$, the computed determinant satisfies

$$\widehat{d} := fl(\det(H)) = (1 + \theta_n) \det(T)\big(\eta - (h + \Delta h)^T (T + \Delta T)^{-1} y\big),$$

where $|q_n| \le \gamma_n$. This is not a backward error result, because only one of the two Ts in this expression is perturbed. However, we can write $\det(T) = \det(T + DT)(1 + q_{(n-1)^2})$, so that

$$\widehat{d} = \det(T + \Delta T)\big(\eta(1 + \theta_{n^2-n+1}) - (h + \Delta h)^T (T + \Delta T)^{-1} y (1 + \theta_{n^2-n+1})\big).$$

We conclude that the computed determinant is the exact determinant of a perturbed Hessenberg matrix in which each element has undergone a relative perturbation not exceeding $\gamma_{n^2-n+1} \approx n^2 u$, which is a very satisfactory result. In fact, the constant γ_{n^2-n+1} may be reduced to γ_{2n-1} by a slightly modified analysis; see Problem 13.14.

13.6. Notes and References

Classic references on matrix inversion are Wilkinson's paper *Error Analysis of Direct Methods of Matrix Inversion* [1085, 1961] and his book *Rounding Errors in Algebraic Processes* [1088, 1963]. In discussing Method 1 of §13.2.1, Wilkinson says "*The residual of X as a left-hand inverse may be larger than the residual as a right-hand inverse by a factor as great as* $\|L\| \, \|L^{-1}\|$ . . . We are asserting that the computed X is *almost invariably* of such a nature that $XL - I$ is equally small" [1088, 1963, p. 107]. Our experience concurs with the latter statement. Triangular matrix inversion provides a good example of the value of rounding error analysis: it helps us identify potential instabilities, even if they are rarely manifested in practice, and it shows us what kind of stability is guaranteed to hold.

In [1085, 1961] Wilkinson identifies various circumstances in which triangular matrix inverses are obtained to much higher accuracy than the bounds of this chapter suggest. The results of §8.2 provide some insight. For example, if T is a triangular M-matrix then, as noted after Corollary 8.10, its inverse is computed to high relative accuracy, no matter how ill conditioned L may be.

Sections 13.2 and 13.3 are based closely on Du Croz and Higham [322, 1992].

Method D in §13.3.4 is used by the Hewlett-Packard HP-15C calculator, for which the method's lack of need for extra storage is an important property [523, 1982].

Higham [560, 1995] gives error analysis of a divide-and-conquer method for inverting a triangular matrix that has some attractions for parallel computation.

GJE is an old method. It was discovered independently by the geodesist Wilhelm Jordan (1842-1899) (not the mathematician Camille Jordan (1838-1922)) and B.-I. Clasen [12, 1987].

An Algol routine for inverting positive definite matrices by GJE was published in the *Handbook for Automatic Computation* by Bauer and Reinsch [83, 1971]. As a means of solving a single linear system, GJE is 50% more expensive than GE when cost is measured in flops; the reason is that GJE takes $O(n^3)$ flops to solve the upper triangular system that GE solves in n^2 flops. However, GJE has attracted interest for vector computing because it maintains full vector lengths throughout the algorithm. Hoffmann [577, 1987] found that it was faster than GE on the CDC Cyber 205, a now-defunct machine with a relatively large vector startup overhead.

Turing [1027, 1948] gives a simplified analysis of the propagation of errors in GJE, obtaining a forward error bound proportional to $\kappa(A)$. Bauer [80, 1966] does a componentwise forward error analysis of GJE for matrix inversion and obtains a relative error bound proportional to $\kappa_\infty(A)$ for symmetric positive definite A. Bauer's paper is in German and his analysis is not easy to follow. A summary of Bauer's analysis (in English) is given by Meinguet [746, 1969].

The first thorough analysis of the stability of GJE was by Peters and Wilkinson [828, 1975]. Their paper is a paragon of rounding error analysis. Peters and Wilkinson observe the connection with GE, then devote their attention to the "second stage" of GJE, in which $Ux = y$ is solved. They show that each component of x is obtained by solving a lower triangular system, and they deduce that, for each i, $(U + \Delta U_i)x^{(i)} = y + \Delta y_i$, where $|\Delta U_i| \le \gamma_n|U|$ and $|\Delta y_i| \le \gamma_n|y|$, and where the ith component of $x^{(i)}$ is the ith component of $\widehat{x}$. They then show that $\widehat{x}$ has relative error bounded by $2n\kappa_\infty(U)u+O(u^2)$, but that $\widehat{x}$ does not necessarily have a small backward error. The more direct approach used in our analysis is similar to that of Dekker and Hoffmann [276, 1989], who give a normwise analysis of a variant of GJE that uses row pivoting (elimination across rows) and column interchanges. Our componentwise bounds (13.29)–(13.32) are new.

Goodnight [473, 1979] describes the use of GJE in statistics for solving least squares problems.

Error analysis of Hyman's method is given by Wilkinson [1084, 1960], [1088, 1963, pp. 147-154], [1089, 1965, pp. 426-431]. Although it dates from the 1950s, Hyman's method is not obsolete: it has found use in methods designed for high-performance computers; see Ward [1065, 1976], Li and Zeng [701, 1992], and Dubrulle and Golub [324, 1994].

13.6.1. LAPACK

Routine `xGETRI` computes the inverse of a general square matrix by Method B using an LU factorization computed by `xGETRF`. The corresponding routine for a symmetric positive definite matrix is `xPOTRI`, which uses Method D, with a Cholesky factorization computed by `xPOTRF`. Inversion of a symmetric indefinite matrix is done by `xSYTRI`. Triangular matrix inversion is done by `xTRTRI`, which uses Method 2C. None of the LAPACK routines compute the determinant, but it is easy for the user to evaluate it after computing an LU factorization.

Problems

13.1. Reflect on this cautionary tale told by Acton [4, 1970, p. 246].

"It was 1949 in Southern California. Our computer was a very new CPC (model 1, number 1) —a 1-second-per-arithmetic-operation clunker that was holding the computational fort while an early electronic monster was being coaxed to life in an adjacent room, From a nearby aircraft company there arrived one day a 16 × 16 matrix of 10-digit numbers whose inverse was desired ... We labored for two days and, after the usual number of glitches that accompany any strange procedure involving repeated handling of intermediate decks of data cards, we were possessed of an inverse matrix. During the checking operations . . . it was noted that, to eight significant figures, the inverse was the transpose of the original matrix! A hurried visit to the aircraft company to explore the source of the matrix revealed that each element had been laboriously hand computed from some rather simple combinations of sines and cosines of a common angle. It took about 10 minutes to prove that the matrix was, indeed, orthogonal!"

13.2. Rework the analysis of the methods of §13.2.2 using the assumptions (12.3) and (12.4), thus catering for possible use of fast matrix multiplication techniques.

13.3. Show that for any nonsingular matrix A,

$$\kappa(A) \geq \max\left\{\frac{\|AX - I\|}{\|XA - I\|}, \frac{\|XA - I\|}{\|AX - I\|}\right\}.$$

This inequality shows that the left and right residuals of X as an approximation to A^{-1} can differ greatly only if A is ill conditioned.

13.4. (Mendelssohn [748, 1956]) Find parametrized 2 × 2 matrices A and X such that the ratio $\|AX - I\|/\|XA - I\|$ can be made arbitrarily large.

13.5. Let X and Y be approximate inverses of $A \in \mathbb{R}^{n\times n}$ that satisfy

$$|A\widehat{X} - I| \leq u|A||\widehat{X}| \quad \text{and} \quad |\widehat{Y}A - I| \leq u|\widehat{Y}||A|,$$

and let $\widehat{x} = fl(\widehat{X}b)$ and $\widehat{y} = fl(\widehat{Y}b)$. Show that

$$|A\widehat{x} - b| \leq \gamma_{n+1}|A||\widehat{X}||b| \quad \text{and} \quad |A\widehat{y} - b| \leq \gamma_{n+1}|A||\widehat{Y}||A||x|.$$

Derive forward error bounds for $\widehat{x}$ and $\widehat{y}$. Interpret all these bounds.

13.6. What is the relation between the matrix on the front cover of the *LAPACK Users' Guide* [17, 1995] and that on the back cover? Answer the same question for the *LINPACK Users' Guide* [307, 1979].

13.7. Show that for any matrix having a row or column of 1s, the elements of the inverse sum to 1.

13.8. Let $X = A + iB \in \mathbb{C}^{n\times n}$ be nonsingular. Show that X^{-1} can be expressed in terms of the inverse of the real matrix of order $2n$,

$$Y = \begin{bmatrix} A & -B \\ B & A \end{bmatrix}.$$

Show that if X is Hermitian positive definite then Y is symmetric positive definite. Compare the economics of real versus complex inversion.

13.9. For a given nonsingular $A \in \mathbb{R}^{n\times n}$ and $X \approx A^{-1}$, it is interesting to ask what is the smallest ε such that $X + DX = (A + DA)^{-1}$ with $\|DX\| \leq \epsilon\|X\|$ and $\|\Delta A\| \leq \epsilon\|A\|$. We require $(A + DA)(X + DX) = I$, or

$$ADX + DAX + DADX = I - AX.$$

It is reasonable to drop the second-order term, leaving a generalized Sylvester equation that can be solved using singular value decompositions of A and X (cf. §15.2). Investigate this approach computationally for a variety of A and methods of inversion.

13.10. For a nonsingular $A \in \mathbb{R}^{n\times n}$ and given integers i and j, under what conditions is $\det(A)$ independent of a_{ij}? What does this imply about the suitability of $\det(A)$ for measuring conditioning?

13.11. Prove Hadamard's inequality for $A \in \mathbb{C}^{n\times n}$:

$$|\det(A)| \le \prod_{k=1}^{n} \|a_k\|_2,$$

where $a_k = A(:, k)$. When is there equality? (Hint: use the QR factorization.)

13.12. (a) Show that if $A^T = QR$ is a QR factorization then the Hadamard condition number $\psi(A) = \prod_{i=1}^{n} \rho_i/|r_{ii}|$, where $\rho_i = \|R(:, i)\|_2$. (b) Evaluate $\psi(A)$ for $A = U(1)$ (see (8.2)) and for the Pei matrix $A = (a - 1)I + ee^T$.

13.13. (Guggenheimer, Edelman, and Johnson [486, 1995]) (a) Prove that for a nonsingular $A \in \mathbb{R}^{n\times n}$,

$$\kappa_2(A) < \frac{2}{|\det(A)|}\left(\frac{\|A\|_F}{n^{1/2}}\right)^n.$$

(Hint: apply the arithmetic-geometric mean inequality to the n numbers $\sigma_1^2/2$, $\sigma_1^2/2$, σ_2^2, $\dots$, σ_{n-1}^2, where the s_i are the singular values of A.) (b) Deduce that if A has rows of unit 2-norm then

$$\kappa_2(A) < \frac{2}{|\det(A)|} = 2\psi(A),$$

where ψ is the Hadamard condition number.

13.14. Show that Hyman's method for computing $\det(H)$, where $H \in \mathbb{R}^{n\times n}$ is an unreduced upper Hessenberg matrix, computes the exact determinant of $H + \Delta H$ where $|\Delta H| \le \gamma_{2n-1}|H|$, barring underflow and overflow. What is the effect on the error bound of a diagonal similarity transformation $H^{r} \longrightarrow D^{-1}HD$, where $D = \mathrm{diag}(d_i)$, $d_i \ne 0$?

13.15. What is the condition number of the determinant?

13.16. (RESEARCH PROBLEM) Obtain backward and forward error bounds for GJE applied to a diagonally dominant matrix. Peters and Wilkinson [828, 1975] state that "it is well known that Gauss–Jordan is stable" for a diagonally dominant matrix, but a proof does not seem to have been published.

Chapter 14
Condition Number Estimation

Most of LAPACK'S condition numbers and error bounds are based on estimated condition numbers . . .
The price one pays for using an estimated rather than an exact condition number is occasional (but very rare) underestimates of the true error; years of experience attest to the reliability of our estimators, although examples where they badly underestimate the error can be constructed.

— E. ANDERSON et al,, *LAPACK Users' Guide, Release 2.0 (1995)*

The importance of the counter-examples is that they make clear that any effort toward proving that the algorithms always produce useful estimations is fruitless.
It may be possible to prove that the algorithms produce useful estimations in certain situations, however, and this should be pursued.
An effort simply to construct more complex algorithms is dangerous.

— A. K. CLINE and R. K. REW, *A Set of Counter-Examples to Three Condition Number Estimators (1983)*

Singularity is almost invariably a clue.

— SIR ARTHUR CONAN DOYLE, *The Boscombe Valley Mystery "(1892)*

14.1. How to Estimate Componentwise Condition Numbers

When bounding the forward error of a computed solution to a linear system we would like to obtain the bound with an order of magnitude less work than is required to compute the solution. For a dense $n \times n$ system, where the solution process usually requires $O(n^3)$ operations, we need to compute the bound in $O(n^2)$ operations. An estimate of the bound that is correct to within a factor 10 is usually acceptable, because it is the magnitude of the error that is of interest, not its precise value.

In the perturbation theory of Chapter 7 for linear equations we obtained perturbation bounds that involve the condition numbers

$$\kappa_{E,f}(A,x) = \frac{\|A^{-1}\|\,\|f\|}{\|x\|} + \|A^{-1}\|\,\|E\|,$$

$$\mathrm{cond}_{E,f}(A,x) = \frac{\|\,|A^{-1}|f + |A^{-1}|E|x|\,\|_\infty}{\|x\|_\infty},$$

and their variants. To compute these condition numbers exactly we need to compute A^{-1}, which requires $O(n^3)$ operations, even if we are given a factorization of A. Is it possible to produce reliable estimates of both condition numbers in $O(n^2)$ operations? The answer is yes, but to see why we first need to rewrite the expression for $\mathrm{cond}_{E,f}(A, x)$. Consider the general expression $\|\,|A^{-1}|d\,\|_\infty$, where d is a given nonnegative vector (thus, $d = f + E|x|$ for $\mathrm{cond}_{E,f}(A, x)$); note that the practical error bound (7.27) is of this form. Writing $D = \mathrm{diag}(d)$ and $e = [1, 1, \ldots, 1]^T$, we have

$$\|\,|A^{-1}|d\,\|_\infty = \|\,|A^{-1}|De\,\|_\infty = \|\,|A^{-1}D|e\,\|_\infty = \|\,|A^{-1}D|\,\|_\infty = \|A^{-1}D\|_\infty. \tag{14.1}$$

Hence the problem is equivalent to that of estimating $\|B\|_\infty := \|A^{-1}D\|_\infty$. If we can estimate this quantity then we can estimate any of the condition numbers or perturbation bounds for a linear system. There is an algorithm, described in §14.3, that produces reliable order-of-magnitude estimates of $\|B\|_1$, for an arbitrary B, by computing just a few matrix-vector products Bx and B^Ty for carefully selected vectors x and y and then approximating $\|B\|_1 \approx \|Bx\|_1/\|x\|_1$. If we assume that we have a factorization of A (say, $PA = LU$ or $A = QR$), then we can form the required matrix-vector products for $B = A^{-1}D$ in $O(n^2)$ flops. Since $\|B\|_1 = \|B^T\|_\infty$, it follows that we can use the algorithm to estimate $\|\,|A^{-1}|d\,\|_\infty$ in $O(n^2)$ flops.

Before presenting the 1-norm condition estimation algorithm, we describe a more general method that estimates $\|B\|_p$ for any $p \in [1,\infty]$. This more general method is of interest in its own right and provides insight into the special case $p = 1$.

14.2. The p-Norm Power Method

An iterative "power method" for computing $\|A\|_p$ was proposed by Boyd in 1974 [139, 1974]. When $p = 2$ it reduces to the usual power method applied to A^TA. We use the notation $\mathrm{dual}_p(x)$ to denote any vector y of unit q-norm such that equality holds in the Holder inequality $x^Ty \le \|x\|_p\|y\|_q$ (this normalization is different from the one we used in (6.3), but is more convenient for our present purposes). Throughout this section, $p \ge 1$ and q is defined by $p^{-1} + q^{-1} = 1$.

Algorithm 14.1 (p-norm power method). Given $A \in \mathbb{R}^{m\times n}$ and $x_0 \in \mathbb{R}^n$, this algorithm computes γ and x such that $\gamma \le \|A\|_p$ and $\|Ax\|_p = \gamma\|x\|_p$.

```
$x = x_0/\|x_0\|_p$
repeat
    $y = Ax$
    $z = A^T \mathrm{dual}_p(y)$
    if $\|z\|_q \le z^Tx$
        $\gamma = \|y\|_p$
        quit
    end
    $x = \mathrm{dual}_q(z)$
end
```

Cost: $4rmn$ flops (for r iterations).

There are several ways to derive Algorithm 14.1. Perhaps the most natural way is to examine the optimality conditions for maximization of

$$F(x) = \frac{\|Ax\|_p}{\|x\|_p}, \quad x \in \mathbb{R}^n.$$

First, we note that the subdifferential (that is, the set of subgradients) of an arbitrary vector norm $\|\cdot\|$ is given by [378, 1987, p. 379]

$$\partial\|x\| = \{\,\lambda : \lambda^Tx = \|x\|,\ \|\lambda\|_D \le 1\,\}.$$

If $x \ne 0$ then $\lambda^Tx = \|x\| \Rightarrow \|\lambda\|_D \ge 1$, from the Holder inequality, and so, if $x \ne 0$,

$$\begin{aligned}\partial\|x\| &= \{\,\lambda : \lambda^Tx = \|x\|,\ \|\lambda\|_D = 1\,\}\\ &=: \{\mathrm{dual}(x)\}.\end{aligned}$$

It can also be shown that if A has full rank,

$$\partial\|Ax\| = \{A^T\,\mathrm{dual}(Ax)\}. \tag{14.2}$$

We assume now that A has full rank, $1 < p < \infty$, and $x \neq 0$. Then it is easy to see that there is a unique vector $\mathrm{dual}_p(x)$, so $\partial\|x\|_p$ has just one element, that is, $\|x\|_p$ is differentiable. Hence we have

$$F(x) = \frac{A^T \mathrm{dual}_p(Ax)}{\|x\|_p} - \frac{\|Ax\|_p \, \mathrm{dual}_p(x)}{\|x\|_p^2}. \tag{14.3}$$

The first-order Kuhn–Tucker condition for a local maximum of F is therefore

$$A^T \mathrm{dual}_p(Ax) = \frac{\|Ax\|_p}{\|x\|_p} \mathrm{dual}_p(x).$$

Since $\mathrm{dual}_q(\mathrm{dual}_p(x)) = x/\|x\|_p$ if $p \neq 1, \infty$, [1] this equation can be written

$$x = \frac{\|x\|_p^2}{\|Ax\|_p} \mathrm{dual}_q(A^T \mathrm{dual}_p(Ax)). \tag{14.4}$$

The power method is simply functional iteration applied to this transformed set of Kuhn–Tucker equations (the scale factor $\|x\|_p^2/\|Ax\|_p$ is irrelevant since $F(ax) = F(x)$).

For the 1- and ∞-norms, which are not everywhere differentiable, a different derivation can be given. The problem can be phrased as one of maximizing the convex function $F(x) := \|Ax\|_p$ over the convex set $S := \{x : \|x\|_p \leq 1\}$. The convexity of F and S ensures that, for any $u \in S$, at least one vector g exists such that

$$F(v) \geq F(u) + g^T(v - u) \quad \text{for all } v \in S. \tag{14.5}$$

Vectors g satisfying (14.5) are called *subgradients* of F (see, for example, [378, 1987, p. 364]). Inequality (14.5) suggests the strategy of choosing a subgradient g and moving from u to a point $v_* \in S$ that maximizes $g^T(v - u)$, that is, a vector that maximizes $g^T v$. Clearly, $v_* = \mathrm{dual}_q(g)$. Since, from (14.2), g has the form $A^T \mathrm{dual}_p(Ax)$, this completes the derivation of the iteration.

For all values of p the power method has the desirable property of generating an increasing sequence of norm approximations.

Lemma 14.2. *In Algorithm 14.1, the vectors from the kth iteration satisfy*
(i) ${z^k}^T x^k = \|y^k\|_p$, *and*
(ii) $\|y^k\|_p \leq \|z^k\|_q \leq \|y^{k+1}\|_p \leq \|A\|_p$.
The first inequality in (ii) *is strict if convergence is not obtained on the kth iteration.*

Proof. ${z^k}^T x^k = \mathrm{dual}_p(y^k)^T A x^k = \mathrm{dual}_p(y^k)^T y^k = \|y^k\|_p$. Then

$$\begin{aligned}\|y^k\|_p &= {z^k}^T x^k\\ &\le \|z^k\|_q \|x^k\|_p = \|z^k\|_q = {z^k}^T x^{k+1} = \mathrm{dual}_p(y^k)^T A x^{k+1}\\ &\le \|\,\mathrm{dual}_p(y^k)\|_q \|Ax^{k+1}\|_p = \|y^{k+1}\|_p\\ &\le \|A\|_p.\end{aligned}$$

For the last part, note that, in view of (i), the convergence test "$\|z^k\|_q \le {z^k}^T x^k$" can be written as "$\|z^k\|_q \le \|y^k\|_p$". □

It is clear from Lemma 14.2 (or directly from the Holder inequality) that the convergence test "$\|z\|_q \le z^T x$" in Algorithm 14.1 is equivalent to "$\|z\|_q = z^T x$" and, since $\|x\|_p = 1$, this is equivalent to $x = \mathrm{dual}_q(z)$. Thus, although the convergence test compares two scalars, it is actually testing for equality in the vector equation (14.4).

The convergence properties of Algorithm 14.1 merit a careful description. First, in view of Lemma 14.2, the scalars $\gamma_k = \|y^k\|_p$ form an increasing and convergent sequence. This does not necessarily imply that Algorithm 14.1 converges, since the algorithm tests for convergence of the x_k, and these vectors could fail to converge. However, a subsequence of the x^k must converge to a limit, $\overline{x}$ say. Boyd [139, 1974] shows that if $\overline{x}$ is a strong local maximum of F with no zero components, then $x^k \to \overline{x}$ linearly.

If Algorithm 14.1 converges it converges to a stationary point of $F(x)$ when $1 < p < \infty$. Thus, instead of the desired global maximum $\|A\|_p$, we may obtain only a local maximum or even a saddle point. When $p = 1$ or ∞, if the algorithm converges to a point at which F is not differentiable, that point need not even be a stationary point. On the other hand, for $p = 1$ or ∞ Algorithm 14.1 terminates in at most $n + 1$ iterations (assuming that when dual_p or dual_q is not unique an extreme point of the unit ball is taken), since the algorithm moves between the vertices e_i of the unit ball in the 1-norm, increasing F on each stage ($x = \pm e_i$ for $p = 1$, and $\mathrm{dual}_p(y) = \pm e_i$ for $p = \infty$). An example where n iterations are required for $p = 1$ is given in Problem 14.2.

For two special types of matrix, more can be said about Algorithm 14.1.

(1) If $A = xy^T$ (rank 1), the algorithm converges on the second step with $\gamma = \|A\|_p = \|x\|_p \|y\|_q$, whatever x_0.

(2) Boyd [139, 1974] shows that if A has nonnegative elements, $A^T A$ is irreducible, $1 < p < \infty$, and x_0 has positive elements, then the x^k converge and $\gamma_k \to \|A\|_p$.

For values of p strictly between 1 and ∞ the convergence behaviour of Algorithm 14.1 is typical of a linearly convergent method: exact convergence is not usually obtained in a finite number of steps and arbitrarily many steps can be required for convergence, as is well-known for the 2-norm power method. Fortunately, there is a method for choosing a good starting vector that can be combined with Algorithm 14.1 to produce a reliable norm estimator; see the Notes and References and Problem 14.1.

We now turn our attention to the extreme values of p: 1 and ∞.

14.3. LAPACK l-Norm Estimator

Algorithm 14.1 has two remarkable properties when p = 1: it almost always converges within four iterations (when $x_0 = [1, 1, \ldots, 1]^T$, say) and it frequently yields $\|A\|_1$ exactly. This rapid, finite termination is also obtained for $p = \infty$, and is related to the fact that Algorithm 14.1 moves among the finite set of extreme points of the unit ball. Numerical experiments suggest that the accuracy is about the same for both norms but that slightly more iterations are required on average for $p = \infty$. Hence we will confine our attention to the 1-norm.

The l-norm version of Algorithm 14.1 was derived independently of the general algorithm by Hager [492, 1984] and can be expressed as follows. The notation $x = \mathrm{sign}(y)$ means that $x_i = 1$ or -1 according as $y_i \geq 0$ or $y_i < 0$. We now specialize to square matrices.

Algorithm 14.3 (1-norm power method). Given $A \in \mathbb{R}^{n\times n}$ this algorithm computes γ and x such that $\gamma \leq \|A\|_1$ and $\|Ax\|_1 = \gamma\|x\|_1$.

```
$x = n^{-1}e$
repeat
    $y = Ax$
    $x = \mathrm{sign}(y)$
    $z = A^Tx$
    if $\|z\|_\infty \leq z^Tx$
        $\gamma = \|y\|_1$
        quit
    end
    $x = e_j$, where $|z_j| = \|z\|_\infty$ (smallest such $j$)
end
```

Numerical experiments show that the estimates produced by Algorithm 14.3 are frequently exact ($\gamma = \|A\|_1$), usually "acceptable" ($\gamma \geq \|A\|_1/10$), and sometimes poor ($\gamma < \|A\|_1/10$).

An important question for any norm or condition estimator is whether there exists a "counterexample"—a parametrized matrix for which the quotient "estimate divided by true norm" can be made arbitrarily small (or large, depending on whether the estimator produces a lower bound or an upper bound) by varying a parameter. A general class of counterexample for Algorithm 14.3 is given by the matrices

$$A = I + \theta C,$$

where $Ce = C^Te = 0$ (there are many possible choices for C). For any such matrix, Algorithm 14.3 computes $y = n^{-1}e$, $x = e$, $z = e$, and hence the algorithm terminates at the end of the first iteration with

$$\frac{\gamma}{\|A\|_1} = \frac{1}{\|I + \theta C\|_1} \sim \theta^{-1} \quad \text{as } \theta \to \infty.$$

The problem is that the algorithm stops at a local maximum that can differ from the global one by an arbitrarily large factor.

A more reliable and more robust algorithm is produced by the following modifications of Higham [537, 1988].

Definition of estimate. To overcome most of the poor estimates, γ is redefined as

$$\gamma = \max\left\{\|y\|_1, \frac{\|c\|_1}{\|b\|_1}\right\},$$

where

$$c = Ab, \quad b_i = (-1)^{i+1}\left(1 + \frac{i-1}{n-1}\right).$$

The vector b is considered likely to "pick out" any large elements of A in those cases where such elements fail to propagate through to y.

Convergence test. The algorithm is limited to a minimum of two and a maximum of five iterations. Further, convergence is declared after computing x if the new x is the same as the previous one; this event signals that convergence will be obtained on the current iteration and that the next (and final) multiplication A^Tx is unnecessary. Convergence is also declared if the new $\|y\|_1$ is no larger than the previous one. This nonincrease of the norm can happen only in finite precision arithmetic and signals the possibility of a vertex e_j being revisited—the onset of "cycling."

The improved algorithm is as follows. This algorithm is the basis of all the condition number estimation in LAPACK.

Algorithm 14.4 (LAPACK norm estimator). Given $A \in \mathbb{R}^{n\times n}$ this algorithm computes γ and $v = Aw$ such that $\gamma \le \|A\|_1$ with $\|v\|_1/\|w\|_1 = \gamma$ (w is not returned).

```
$v = A(n^{-1}e)$
if $n = 1$, quit with $\gamma = |v_1|$, end
$\gamma = \|v\|_1$
$x = \mathrm{sign}(v)$
```

```
    $x = A^Tx$
    $k = 2$
    repeat
        $j = \min\{i: |x_i| = \|x\|_\infty\}$
        $\upsilon = Ae_j$
        $\overline{\gamma} = \gamma$
        $\gamma = \|\upsilon\|_1$
        if $\mathrm{sign}(\upsilon) = x$ or $\gamma \le \overline{\gamma}$, goto (*), end
        $x = \mathrm{sign}(\upsilon)$
        $x = A^Tx$
        $k = k+1$
    until $(\|x\|_\infty = x_j$ or $k > 5)$
(*) $x_i = (-1)^{i+1}\left(1 + \frac{i-1}{n-1}\right), \quad i = 1\colon n$
    $x = Ax$
    if $2\|x\|_1/(3n) > \gamma$ then
      $\upsilon = x$
      $\gamma = 2\|x\|_1/(3n)$
    end
```

Algorithm 14.4 can still be “defeated”: it returns an estimate 1 for matrices $A(q)$ of the form

$$A(q) = I + qP, \text{ where } P = P^T,\ Pe = 0,\ Pe_1 = 0,\ Pb = 0. \tag{14.6}$$

(P can be constructed as $I - Q$ where Q is the orthogonal projection onto span{e, e_1, b}.) Indeed, the existence of counterexamples is intuitively obvious since Algorithm 14.4 samples the behaviour of A on less than n vectors in $\mathbb{R}^n$. Numerical counterexamples (not parametrized) can be constructed automatically by direct search, as described in §24.3.1. Despite these weaknesses, practical experience with Algorithm 14.4 shows that it is very rare for the estimate to be more than three times smaller than the actual norm, independent of the dimension n. Therefore Algorithm 14.4 is, in practice, a very *reliable* norm estimator. The number of matrix-vector products required is at least 4 and at most 11, and averages between 4 and 5.

There is an analogue of Algorithm 14.3 for complex matrices, in which x_i is defined as $y_i/|y_i|$ if $y_i \neq 0$ and 1 otherwise. In the corresponding version of Algorithm 14.4 the test for repeated x vectors is removed, because x now has noninteger, complex components and so is unlikely to repeat.

It is interesting to look at a performance profile of Algorithm 14.4. A performance profile is a plot of some measure of the performance of an algorithm versus a problem parameter. In this case, the natural measure of performance is the underestimation ratio, $\gamma/\|A\|_1$. Figure 14.1 shows the performance profile for a 5 × 5 matrix $A(q)$ of the form (14.6), with P constructed as

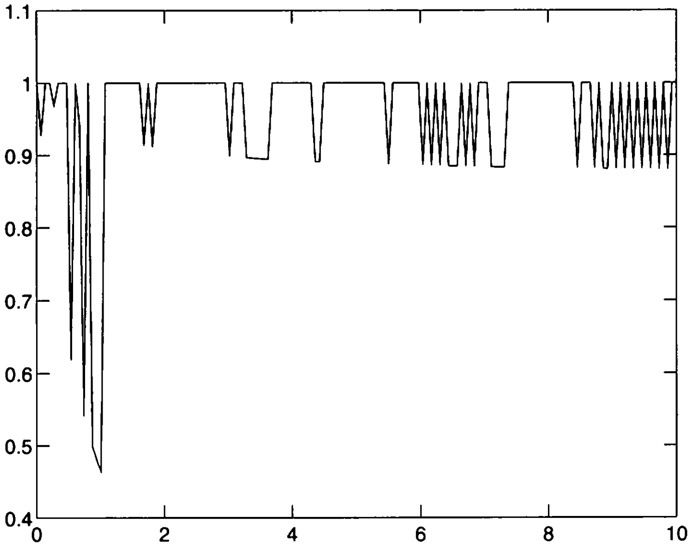


Figure 14.1. *Underestimation ratio for Algorithm* 14.4 *for* 5×5 *matrix A(O) of* (14.6) *with 150 equally spaced values of* $\theta \in [0,10]$.

described above (because of rounding errors in constructing $A(q)$ and within the algorithm, the computed norm estimates differ from those that would be produced inexact arithmetic). The jagged nature of the performance curve is typical for algorithms that contain logical tests and branches. Small changes in the parameter q, which themselves result in different rounding errors, can cause the algorithm to visit different vertices in this example.

14.4. Other Condition Estimators

The first condition estimator to be widely used is the one employed in LINPACK. It was developed by Cline, Moler, Stewart, and Wilkinson [216, 1979]. The idea behind this condition estimator originates with Gragg and Stewart [476, 1976], who were interested in computing an approximate null vector rather than estimating the condition number itself.

We will describe the algorithm as it applies to a triangular matrix $T \in \mathbb{R}^{n\times n}$. There are three steps:

1. Choose a vector d such that $\|y\|$ is as large as possible relative to $\|d\|$, where $T^Ty = d$.
2. Solve $Tx = y$.

3. Estimate $\|T^{-1}\| \approx \|x\|/\|y\| \; (\le \|T^{-1}\|)$.

In LINPACK the norm is the 1-norm, but the algorithm can also be used for the 2-norm or the cm-norm. The motivation for step 2 is based on a singular value decomposition analysis. Roughly, if $\|y\|/\|d\| \; (\approx \|T^{-T}\|)$ is large then $\|x\|/\|y\| \; (\approx \|T^{-1}\|)$ will almost certainly be at least as large, and it could be a much better estimate. Notice that $T^TTx = d$, so the algorithm is related to the power method on the matrix $(T^TT)^{-1}$ with the specially chosen starting vector d.

To examine step 1 more closely, suppose that $T = U^T$ is lower triangular and note that the equation $Uy = d$ can be solved by the following column-oriented (saxpy) form of substitution:

```
      $p(1{:}n) = 0$
      for $j = n{:}\,-1{:}\,1$
          $y_j = (d_j - p_j)/u_{jj}$
(*)       $p(1{:}j-1) = p(1{:}j-1) + U(1{:}j-1,j)y(j)$
      end
```

The idea is to choose the elements of the right-hand side vector d adaptively as the solution proceeds, with $d_j = \pm 1$. At the jth stage of the algorithm d . . . , d_{j+1} have been chosen and y_n, . . . , y_{j+1} are known. The next element $d_j \in \{+1, -1\}$ is chosen so as to maximize a weighted sum of $d_j - p_j$ and the partial sums p_1, . . . , p_j, *which* would be computed during the next execution of statement (*) above. Hence the algorithm looks ahead, trying to gauge the effect of the choice of d_j on future solution components. This heuristic algorithm for choosing d is expressed in detail as follows.

Algorithm 14.5 (LINPACK condition estimator). Given a nonsingular upper triangular matrix $U \in \mathbb{R}^{n\times n}$ and a set of nonnegative weights $\{w_i\}$, this algorithm computes a vector y such that $Uy = d$, where the elements $d_j = \pm 1$ are chosen to make $\|y\|$ large.

```
$p(1{:}n) = 0$
for $j = n{:}\,-1{:}\,1$
    $y_j^+ = (1 - p_j)/u_{jj}$
    $y_j^- = (-1 - p_j)/u_{jj}$
    $p^+(1{:}j-1) = p(1{:}j-1) + U(1{:}j-1,j)y^+(j)$
    $p^-(1{:}j-1) = p(1{:}j-1) + U(1{:}j-1,j)y^-(j)$
    if $w_j|1 - p_j| + \sum_{i=1}^{j-1} w_i|p_i^+| \ge w_j|1 + p_j| + \sum_{i=1}^{j-1} w_i|p_i^-|$
       $y_j = y_j^+$
       $p(1{:}j-1) = p^+(1{:}j-1)$
    else
       $y_j = y_j^-$
```

```
        $p(1{:}j-1) = p^-(1{:}j-1)$
    end
end
```

cost: $4n^2$ flops.

LINPACK takes the weights $w_j \equiv 1$, though another possible (but more expensive) choice would be $w_j = 1/|u_{jj}|$, which corresponds to how p_j is weighted in the expression $y_j = (d_j - p_j)/u_{jj}$.

To estimate $\|A^{-1}\|$ for a full matrix A, the LINPACK estimator makes use of an LU factorization of A. Given $PA = LU$, the equations solved are $U^Tz = d$, $L^Ty = z$, and $AX = P^Ty$, where for the first system d is constructed by the analogue of Algorithm 14.5 for lower triangular matrices; the estimate is $\|x\|_1/\|y\|_1 \approx \|A^{-1}\|_1$. Since d is chosen without reference to L, there is an underlying assumption that any ill condition in A is reflected in U. This assumption may not be true; see Problem 14.3.

In contrast to the LAPACK norm estimator, the LINPACK estimator requires explicit access to the elements of the matrix. Hence the estimator cannot be used to estimate componentwise condition numbers. Furthermore, separate code has to be written for each different type of matrix and factorization. Consequently, while LAPACK has just a single norm estimation routine, which is called by many other routines, LINPACK has multiple versions of its algorithm, each tailored to the specific matrix or factorization.

Several years after the LINPACK condition estimator was developed, several parametrized counterexamples were found by Cline and Rew [217, 1983]. Numerical counterexamples can also be constructed by direct search, as shown in §24.3.1. Despite the existence of these counterexamples the LINPACK estimator has been widely used and is regarded as being almost certain to produce an estimate correct to within a factor 10 in practice.

A 2-norm condition estimator was developed by Cline, Corm, and Van Loan [218, 1982, Algorithm 1]; see also Van Loan [1043, 1987] for another explanation. The algorithm builds on the ideas underlying the LINPACK estimator by using "look-behind" as well as look-ahead. It estimates $\sigma_{\min}(R) = \|R^{-1}\|_2^{-1}$ or $\sigma_{\max}(R) = \|R\|_2$ for a triangular matrix R, where $\sigma_{\min}$ and $\sigma_{\max}$ denote the smallest and largest singular values, respectively. Full matrices can be treated if a factorization $A = QR$ is available (Q orthogonal, R upper triangular), since R and A have the same singular values. The estimator performs extremely well in numerical tests, often producing an estimate that has some correct digits [218, 1982], [534, 1987]. No counterexamples to the estimator were known until Bischof [103, 1990] obtained counterexamples as a by-product of the analysis of a different but related method, mentioned at the end of this section.

All the methods described so far have the property that when applied repeatedly to a given matrix they always produce the same estimate. Another approach is to introduce some randomness, so that the output of the method depends on the particular random numbers chosen. A natural idea along these lines is to apply the power method to the matrix $(AA^T)^{-1}$ with a randomly chosen starting vector. If a factorization of A is available, the power method vectors can be computed inexpensively by solving linear systems with A and A^T. Analysis based on the singular value decomposition suggests that there is a high probability that a good estimate of $\|A^{-1}\|_2$ will be obtained. This notion is made precise by Dixon [306, 1983], who proves the following result.

Theorem 14.6 (Dixon). *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular and let $q > 1$ be a constant. If $x \in \mathbb{R}^n$ is a random vector from the uniform distribution on the unit sphere $S_n = \{y \in \mathbb{R}^n : y^Ty = 1\}$, then the inequality*

$$\left(x^T(AA^T)^{-k}x\right)^{1/2k} \le \|A^{-1}\|_2 \le \theta\left(x^T(AA^T)^{-k}x\right)^{1/2k} \qquad (14.7)$$

holds with probability at least $1 - 0.8q^{-k/2}n^{1/2}$ $(k \ge 1)$. □

Note that the left-hand inequality in (14.7) always holds; it is only the right-hand inequality that is in question.

For $k = 1$, (14.7) can be written as

$$\|A^{-1}x\|_2 \le \|A^{-1}\|_2 \le \theta\|A^{-1}x\|_2,$$

which suggests the simple estimate $\|A^{-1}\|_2 \approx \|A^{-1}x\|_2$, where x is chosen randomly from the uniform distribution on S_n. Such vectors x can be generated from the formula

$$x_i = z_i/\|z\|_2,$$

where $z_1, \ldots, z_n$ are independent random variables from the normal $N(0,1)$ distribution [668, 1981, p. 130]. If, for example, $n = 100$ and q has the rather large value 6400 then inequality (14.7) holds with probability at least 0.9.

In order to take a smaller constant q, for fixed n and a desired probability, we can use larger values of k. If $k = 2j$ is even then we can simplify (14.7), obtaining

$$\|(AA^T)^{-j}x\|_2^{1/2j} \le \|A^{-1}\|_2 \le \theta\|(AA^T)^{-j}x\|_2^{1/2j}, \qquad (14.8)$$

and the minimum probability stated by the theorem is $1 - 0.8q^{-j}n^{1/2}$. Taking $j = 3$, for the same value $n = 100$ as before, we find that (14.8) holds with probability at least 0.9 for the considerably smaller value $q = 4.31$.

Probabilistic condition estimation has not yet been adopted in any major software packages, perhaps because the other techniques work so well. For more on the probabilistic power method approach see Dixon [306, 1983], Higham [534, 1987], and Kuczynski and Wozniakowski [676, 1992] (who also analyse the more powerful Lanczos method with a random starting vector). For a probabilistic condition estimation method of very general applicability see Kenney and Laub [652, 1994] and Gudmundsson, Kenney, and Laub [485, 1995].

The condition estimators described above assume that a single estimate is required for a matrix given in its entirety. Condition estimators have also been developed for more specialized situations. Bischof [103, 1990] develops a method for estimating the smallest singular value of a triangular matrix which processes the matrix a row or a column at a time. This "incremental condition estimation" method can be used to monitor the condition of a triangular matrix as it is generated, and so is useful in the context of matrix factorization such as the *QR* factorization with column pivoting. The estimator is generalized to sparse matrices by Bischof, Lewis, and Pierce [104, 1990]. Barlow and Vemulapati [67, 1992] develop a l-norm incremental condition estimator with look-ahead for sparse matrices.

Condition estimates are also required in applications where a matrix factorization is repeatedly updated as a matrix undergoes low rank changes. Algorithms designed for a recursive least squares problem and employing the Lanczos method are described by Ferng, Golub, and Plemmons [372, 1991]. Pierce and Plemmons [831, 1992] describe an algorithm for use with the Cholesky factorization as the factorization is updated, while Shroff and Bischof [918, 1992] treat the *QR* factorization.

14.5. Condition Numbers of Tridiagonal Matrices

For a bidiagonal matrix B, $|B^{-1}| = M(B)^{-1}$ (see §8.3), so the condition numbers $\kappa_{E,f}$ and $\mathrm{cond}_{E,f}$ can be computed exactly with an order of magnitude less work than is required to compute B^{-1} explicitly. This property holds more generally for several types of tridiagonal matrix, as a consequence of the following result. Recall that the LU factors of a tridiagonal matrix are bidiagonal and may be computed using the formulae (9.16).

Theorem 14.7. *If the nonsingular tridiagonal matrix* $A \in \mathbb{R}^{n\times n}$ *has the LU factorization* $A = LU$ *and* $|L||U| = |A|$, *then* $|U^{-1}||L^{-1}| = |A^{-1}|$.

Proof. Using the notation of (9.15), $|L||U| = |A| = |LU|$ if and only if, for all 2,

$$|l_i e_{i-1} + u_i| = |l_i||e_{i-1}| + |u_i|,$$

that is, if

$$\mathrm{sign}\left(\frac{l_i e_{i-1}}{u_i}\right) = 1. \qquad (14.9)$$

Using the formulae

$$(U^{-1})_{ij} = \frac{1}{u_j}\prod_{p=i}^{j-1}\left(\frac{-e_p}{u_p}\right), \qquad j \ge i,$$

$$\left(L^{-1}\right)_{ij} = \prod_{p=j}^{i-1}(-l_{p+1}), \qquad i \geq j,$$

we have

$$\begin{aligned}
\left(U^{-1}L^{-1}\right)_{ij} &= \sum_{k=\max(i,j)}^{n} \left(U^{-1}\right)_{ik}\left(L^{-1}\right)_{kj} \\
&= \sum_{k=\max(i,j)}^{n} \frac{1}{u_k} \prod_{p=i}^{k-1}\left(\frac{-e_p}{u_p}\right)\prod_{p=j}^{k-1}(-l_{p+1}) \\
&= \prod_{p=i}^{\max(i,j)-1}\left(\frac{-e_p}{u_p}\right) \cdot \prod_{p=j}^{\max(i,j)-1}(-l_{p+1}) \\
&\quad \times \sum_{k=\max(i,j)}^{n} \frac{1}{u_k} \prod_{p=\max(i,j)}^{k-1}\left(\frac{e_p l_{p+1}}{u_p}\right) \\
&= \prod_{p=i}^{\max(i,j)-1}\left(\frac{-e_p}{u_p}\right) \cdot \prod_{p=j}^{\max(i,j)-1}(-l_{p+1}) \\
&\quad \times \frac{1}{u_{\max(i,j)}} \cdot \sum_{k=\max(i,j)}^{n} \prod_{p=\max(i,j)}^{k-1}\left(\frac{e_p l_{p+1}}{u_{p+1}}\right).
\end{aligned}$$

Thus, in view of (14.9), it is clear that $|U^{-1}L^{-1}|_{ij} = (|U^{-1}||L^{-1}|)_{zj}$, as required. □

Since L and U are bidiagonal, $|U^{-1}| = M(U)^{-1}$ and $|L^{-1}| = M(L)^{-1}$. Hence, if $|A| = |L||U|$, then, from Theorem 14.7,

$$|A^{-1}|y = |U^{-1}||L^{-1}|y = M(U)^{-1}M(L)^{-1}y. \tag{14.10}$$

It follows that we can compute any of the condition numbers or forward error bounds of interest *exactly* by solving two bidiagonal systems. The cost is $O(n)$ flops, as opposed to the $O(n^2)$ flops needed to compute the inverse of a tridiagonal matrix.

When does the condition $|A| = |L||U|$ hold? Theorem 9.11 shows that it holds if the tridiagonal matrix A is symmetric positive definite, totally positive, or an M-matrix. So for these types of matrix we have a very satisfactory way to compute the condition number.

If A is tridiagonal and diagonally dominant by rows, then we can compute in $O(n)$ flops an upper bound for the condition number that is not more than a factor $2n - 1$ too large.

Theorem 14.8. *Suppose the nonsingular, row diagonally dominant tridiagonal matrix $A \in \mathbb{R}^{n\times n}$ has the LU factorization $A = LU$. Then, if $y \geq 0$,*

$$\|\,|U^{-1}||L^{-1}|y\,\|_\infty \leq (2n-1)\|\,|A^{-1}|y\,\|_\infty.$$

Proof. We have $L^{-1} = UA^{-1}$, so

$$|U^{-1}||L^{-1}|y \leq |U^{-1}||U||A^{-1}|y = |V^{-1}||V||A^{-1}|y,$$

where the bidiagonal matrix $V = \mathrm{diag}(u_{ii})^{-1} U$ has $u_{ii} \equiv 1$ and $|u_{i,i+1}| = |e_i/u_i| \leq 1$ (see the proof of Theorem 9.12). Thus

$$|U^{-1}||L^{-1}|y \leq \begin{bmatrix} 1 & 1 & \dots & 1 \\ & 1 & \dots & 1 \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & & \\ & 1 & \ddots & \\ & & \ddots & 1 \\ & & & 1 \end{bmatrix} |A^{-1}|y,$$

and the result follows on taking norms. $\square$

In fact, it is possible to compute $|A^{-1}|y$ exactly in $O(n)$ operations for any tridiagonal matrix. This is a consequence of the special form of the inverse of a tridiagonal matrix.

Theorem 14.9 (Ikebe). *Let $A \in \mathbb{R}^{n\times n}$ be tridiagonal and irreducible (that is, $a_{i+1,i}$ and $a_{i,i+1}$ are nonzero for all i). Then there are vectors x, y, p, and q such that*

$$(A^{-1})_{ij} = \begin{cases} x_i y_j, & i \leq j, \\ p_i q_j, & i \geq j. \end{cases} \qquad \square$$

This result says that the inverse of an irreducible tridiagonal matrix is the upper triangular part of a rank-1 matrix joined along the diagonal to the lower triangular part of another rank-1 matrix. If A is reducible then it has the block form $\left[\begin{smallmatrix} A_{11} & 0 \\ A_{21} & A_{22} \end{smallmatrix}\right]$ (or its transpose), and this blocking can be applied recursively until the diagonal blocks are all irreducible, at which point the theorem can be applied to the diagonal blocks.

The vectors x, y, p, and q in Theorem 14.9 can all be computed in $O(n)$ flops, and this enables the condition numbers and forward error bounds to be computed also in $O(n)$ flops (see Problem 14.5). Unfortunately, the vectors x, y, p, and q can have a huge dynamic range, causing the computation to break down because of overflow or underflow. For example, for the diagonally dominant tridiagonal matrix with $a_{ii} \equiv 4$, $a_{i+1,i} = a_{i,i+1} \equiv 1$, we have ($x_1 = 1$) $|x_n| \approx q^{n-1}$, $|y_1| \approx q^{-1}$, and $|y_n| \approx q^{-n}$, where $q = 2 + \sqrt{3} \approx 3.73$. These numerical problems can be overcome, but only at a nontrivial increase in cost. Therefore, we do not recommend the use of Theorem 14.9 for computing condition numbers. For a general tridiagonal matrix it is probably better to *estimate* the condition number using Algorithm 14.4.

14.6. Notes and References

The clever trick (14.1) for converting the norm $\| |A^{-1}|d \|_\infty$ into the norm of a matrix with which products are easily formed is due to Arioli, Demmel, and Duff [24, 1989].

The p-norm power method was first derived and analysed by Boyd [139, 1974] and was later investigated by Tao [993, 1984]. Tao applies the method to an arbitrary mixed subordinate norm $||A||_{a,b}$ (see (6.6)), while Boyd takes the a and b-norms to be p-norms (possibly different). Algorithm 14.1 can be converted to estimate $||A||_{a,b}$ by making straightforward modifications to the norm-dependent terms. An algorithm that estimates $||A||_p$ using the power method with a specially chosen starting vector is developed by Higham [551, 1992]; the method for obtaining the starting vector is outlined in Problem 14.1. The estimate produced by this algorithm is always within a factor $n^{1-1/p}$ of $||A||_p$ and the cost is about $70n^2$ flops. A MATLAB M-file pnorm implementing this method is part of the Test Matrix Toolbox (see Appendix E).

The finite convergence of the power method for $p = 1$ and p = 00 holds more generally: if the power method is applied to the norm $||\cdot||_{a,b}$ and one of the a and b norms is polyhedral (that is, its unit ball has a finite number of extreme points), then the iteration converges in a finite number of steps. Moreover, under a reasonable assumption, this number of steps can be bounded in terms of the number of extreme points of the unit balls in the a-norm and the dual of the b-norm. See Bartels [75, 1991] and Tao [993, 1984] for further details.

Hager [492, 1984] gave a derivation of the 1-norm estimator based on subgradients and used the method to estimate $\kappa_1(A)$. That the method is of wide applicability y because it accesses A only through matrix-vector products was recognized by Higham, who developed Algorithm 14.4 and its complex analogue and wrote Fortran 77 implementations, which use a reverse communication interface [537, 1988], [543, 1990]. These codes are used in LAPACK, the NAG library, and various other program libraries. A version of Algorithm 14.4 dedicated to estimating $\kappa_1(A)$ is supplied with MATLAB as M-file condest. Algorithm 14.4 is also implemented in ROM on the Hewlett-Packard HP 48G and HP 48GX calculators (along with several other LAPACK routines), in a form that estimates $\kappa_1(A)$. The Hewlett-Packard implementation is instructive because it shows that condition estimation can be efficient even for small dimensions: on a standard HP 48G, inverting A and estimating its condition number (without being given a factorization of A in either case) both take about 5 seconds for $n = 10$, while for $n = 20$ inversion takes 30 seconds and condition estimation only 20 seconds.

Moler [769, 1978] describes an early version of the LINPACK condition estimator and raises the question of the existence of counterexamples. An early version without look-ahead was incorporated in the Fortran code decomp in the book of Forsythe, Malcolm, and Moler [395, 1977].

Matrices for which condition estimators perform poorly can be very hard to find theoretically or with random testing, but for all the estimators described in this chapter they can be found quite easily by applying direct search optimization to the under- or overestimation ratio; see §24.3.1.

Both LINPACK and LAPACK return estimates of the reciprocal of the condition number, in a variable rcond ≤ 1. Overflow for a very ill conditioned matrix is thereby avoided, and rcond is simply set to zero when singularity is detected. MATLAB has a built-in function rcond that implements the LINPACK condition estimation algorithm.

A simple modification to the LINPACK estimator that can produce a larger estimate is suggested by O'Leary [804, 1980]. For sparse matrices, Grimes and Lewis [483, 1981] suggest a way to reduce the cost of the scaling strategy used in LINPACK to avoid overflow in the condition estimation. Zlatev, Wasniewski, and Schaumburg [1134, 1986] describe their experience in implementing the LINPACK condition estimation algorithm in a software package for sparse matrices.

Stewart [946, 1980] describes an efficient way to generate random matrices of a given condition number and singular value distribution (see §26.3) and tests the LINPACK estimator on such random matrices.

Condition estimators specialized to the (generalized) Sylvester equation have been developed by Byers [172, 1984], Kågström and Westin [624, 1989], and Kågström and Poromaa [621, 1992].

A survey of condition estimators up to 1987, which includes counterexamples and the results of extensive numerical tests, is given by Higham [534, 1987].

Theorems 14.7 and 14.8 are from Higham [541, 1990]. That $\|A^{-1}\|_\infty$ can be computed in $O(n)$ flops for symmetric positive definite tridiagonal A was first, shown in Higham [531, 1986].

Theorem 14.9 has a long history, having been discovered independently in various forms by different authors. The earliest reference we know for the result as stated is Ikebe [600, 1979], where a more general result for Hessenberg matrices is proved. A version of Theorem 14.9 for symmetric tridiagonal matrices was proved by Bukhberger and Emel'yanenko [157, 1973]. The culmination of the many papers on inverses of tridiagonal and Hessenberg matrices is a result of Cao and Stewart on the form of the inverse of a block matrix (A_{ij}) with $A_{ij} = 0$ for $i > j + s$ [184, 1986]; despite the generality of this result, the proof is short and elegant. *Any* banded matrix has an inverse with a special "low rank" structure; the earliest reference on the inverse of a general band matrix is Asplund [32, 1959]. For a recent survey on the inverses of symmetric tridiagonal and block tridiagonal matrices see Meurant [751, 1992].

For symmetric positive definite tridiagonal A the standard way to solve

$AX = b$ is by using a Cholesky or LDL^T factorization, rather than an LU factorization. The LINPACK routine **SPTSL** uses a nonstandard "LUB" factorization resulting from the BABE ("burn at both ends") algorithm, which eliminates from the middle of the matrix to the top and bottom simultaneously (see the LINPACK Users' Guide [307, 1979, Chap. 7] and Higham [531, 1986]). The results of §14.5 are applicable to all these factorization, with minor modifications.

14.6.1. LAPACK

Algorithm 14.4 is implemented in routine `xLACON`, which has a reverse communication interface. The LAPACK routines `xPTCON` and `xPTRFS` for symmetric positive definite tridiagonal matrices compute condition numbers using (14.10); the LAPACK routines `xGTCON` and `xGTRFS` for general tridiagonal matrices use Algorithm 14.4. LINPACK'S tridiagonal matrix routines do not incorporate condition estimation.

Problems

14.1. (Higham [551, 1992]) The purpose of this problem is to develop a non-iterative method for choosing a starting vector for Algorithm 14.1. The idea is to choose the components of x in the order $x_1, x_2, \dots, x_n$ in an attempt to maximize $\|Ax\|_p/\|x\|_p$. Suppose $x_1, \dots, x_{k-1}$ satisfying $\|x(1:k-1)\|_p = 1$ have been determined and let $\gamma_{k-1} = \|A(:, 1:k-1)x(1:k-1)\|_p$. We now try to choose x_k, and at the same time revise $x(1:k-1)$, to give the next partial product a larger norm. Defining

$$g(\lambda,\mu) = \|\lambda A(:,1:k-1)x(1:k-1) + \mu A(:,k)\|_p$$

we set

$$x_k = \mu^*, \qquad x(1:k-1) \leftarrow \lambda^* x(1:k-1),$$

where

$$g(\lambda^*,\mu^*) = \max\left\{ g(\lambda,\mu) : \left\| \begin{bmatrix} \lambda \\ \mu \end{bmatrix} \right\|_p = 1 \right\}.$$

Then $\|x(1:k)\|_p = 1$ and

$$\gamma_k = \|A(:,1:k)x(1:k)\|_p \ge \gamma_{k-1}.$$

Develop this outline into a practical algorithm. What can you prove about the quality of the estimate $\|Ax\|_p/\|x\|_p$ that it produces?

14.2. (Higham [543, 1990]) Let the $n \times n$ symmetric tridiagonal matrix $T_n(a) = (t_{ij})$ be defined by

$$t_{ii} = \begin{cases} 2, & i = 1, \\ i, & 2 \le i \le n-1, \\ -t_{n,n-1} + \alpha, & i = n, \end{cases}$$

$$t_{i,i+1} = \begin{cases} -((i+1)/2 - \alpha) & \text{if } i \text{ is odd}, \\ -i/2 & \text{if } i \text{ is even}. \end{cases}$$

For example, $T_6(a)$ is given by

$$\begin{bmatrix} 2 & -(1-\alpha) & & & & \\ -(1-\alpha) & 2 & -1 & & & \\ & -1 & 3 & -(2-\alpha) & & \\ & & -(2-\alpha) & 4 & -2 & \\ & & & -2 & 5 & -(3-\alpha) \\ & & & & -(3-\alpha) & 3 \end{bmatrix}.$$

Note that, for all a, $\|T_n(a)e_{n-1}\|_1 = \|T_n(a)\|_1$. Show that if Algorithm 14.3 is applied to $T_n(a)$ with $0 \le a < 1$ then $x = e_{i-1}$ on the ith iteration, for $i = 2, \ldots, n$, with convergence on the nth iteration. Algorithm 14.4, however, terminates after five iterations with $y^5 = T_n(a)e_4$, and

$$\frac{\|y^5\|_1}{\|T_n(\alpha)\|_1} = \frac{8-\alpha}{2n-2-\alpha} \rightarrow 0 \quad \text{as } n \rightarrow \infty.$$

Show that the extra estimate saves the day, so that Algorithm 14.4 returns a final estimate that is within a factor 3 of the true norm, for any $a < 1$.

14.3. Let $PA = LU$ be an LU factorization with partial pivoting of A 1 $\mathbb{R}^{n\times n}$. Show that

$$\frac{\|A^{-1}\|_\infty}{2^{n-1}} \le \|U^{-1}\|_\infty \le n\|A^{-1}\|_\infty.$$

14.4. (Higham [537, 1988]) Investigate the behaviour of Algorithms 14.3 and **14.4** for the Pei matrix, $A = aI + ee^T$ ($a \ge 0$), and for the upper bidiagonal matrix with 1s on the diagonal and the first superdiagonal.

14.5. (Ikebe [600, 1979], Higham [531, 1986]) Let $A \in \mathbb{R}^{n\times n}$ be nonsingular, tridiagonal, and irreducible. By equating the last columns in $AA^{-1} = I$ and the first rows in $A^{-1}A = I$, show how to compute the vectors x and y in Theorem 14.9 in $O(n)$ flops. Hence obtain an $O(n)$ flops algorithm for computing $\|\,|A^{-1}|d\,\|_\infty$, where $d \ge 0$.

14.6. The representation of Theorem 14.9 for the inverse of nonsingular, tridiagonal, and irreducible $A \in \mathbb{R}^{n\times n}$ involves 472 parameters, yet A depends only on $3n - 2$ parameters. Obtain an alternative representation that involves only $3n - 2$ parameters. (Hint: symmetrize the matrix.)

14.7. (RESEARCH PROBLEM) (Demmel [286, 1992]) Show that estimating $\|A^{-1}\|$ to within a factor depending only on the dimension of A is at least as expensive as computing A^{-1}.

14.8. (RESEARCH PROBLEM) Let $A \in \mathbb{R}^{n\times n}$ be diagonally dominant by rows, let $A = LU$ be an LU factorization, and let $y \ge 0$. What is the maximum size of $\| |U^{-1}||L^{-1}|y \|_\infty / \|A^{-1}y\|_\infty$ This is an open problem raised in [541, 1990]. In a small number of numerical experiments with full random matrices the ratio has been found to be less than 2 [541, 1990], [790, 1986].

Chapter 15

The Sylvester Equation

We must commence, not with a *square,*
but with an oblong arrangement of terms consisting, suppose,
of m *lines and* n *columns.*
This will not in itself represent a determinant,
but is, as it were, a Matrix out of which we may form
various systems of determinants by fixing upon a number p*,*
and selecting at will p *lines and* p *columns,*
the squares corresponding to which may be termed
determinants of the pth *order.*

— J. J. SYLVESTER, *Additions to the Articles, "On a New Class of Theorems, " and "On Pasta/'s Theorem" (1850)*

I have in previous papers defined a "Matrix" as a rectangular array of terms,
out of which different systems of determinants may be engendered,
as from the womb of a common parent;
these cognate determinants being
by no means isolated in their relations to one another,
but subject to certain simple laws of
mutual dependence and simultaneous deperition.

— J. J. SYLVESTER, *On the Relation Between the Minor Determinants of Linear/y Equivalent Quadratic Functions (1851)*

The linear matrix equation

$$AX - XB = C, \tag{15.1}$$

where $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{n\times n}$ and $C \in \mathbb{R}^{m\times n}$ are given and $X \in \mathbb{R}^{m\times n}$ is to be determined, is called the Sylvester *equation.* It is of pedagogical interest because it includes as special cases several important linear equation problems:

1. linear system: $Ax = c$,
2. multiple right-hand side linear system: $AX = C$,
3. matrix inversion: $AX = I$,
4. eigenvector corresponding to given eigenvalue b: $(A - bI)x = 0$,
5. commuting matrices: $AX - XA = 0$.

The Sylvester equation arises in its full generality in various applications. For example, the equations

$$\begin{bmatrix} I & -X \\ 0 & I \end{bmatrix} \begin{bmatrix} A & -C \\ 0 & B \end{bmatrix} \begin{bmatrix} I & X \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & AX - XB - C \\ 0 & B \end{bmatrix}$$

show that block-diagonalizing a block triangular matrix is equivalent to solving a Sylvester equation. The Sylvester equation can also be produced from finite difference discretization of a separable elliptic boundary value problem on a rectangular domain, where A and B represent application of a difference operator in the "y" and "x" directions, respectively [935, 1991].

That (15.1) is merely a linear system is emphasized by writing it in the form

$$(I_n \otimes A - B^T \otimes I_m)\,\mathrm{vec}(X) = \mathrm{vec}(C), \tag{15.2}$$

where $A \otimes B := (a_{ij}B)$ is a Kronecker product and the vec operator stacks the columns of a matrix into one long vector. For future reference, we note the useful relation

$$\mathrm{vec}(AXB) = (B^T \otimes A)\,\mathrm{vec}(X).$$

(See Horn and Johnson [581, 1991, Chap. 4] for a detailed presentation of properties of the Kronecker product and the vec operator). The $mn \times mn$ coefficient matrix in (15.2) has a very special structure, illustrated for $n = 3$ by

$$\begin{bmatrix} A - b_{11}I & -b_{21}I & -b_{31}I \\ -b_{12}I & A - b_{22}I & -b_{32}I \\ -b_{13}I & -b_{23}I & A - b_{33}I \end{bmatrix}.$$

In dealing with the Sylvester equation it is vital to consider this structure and not treat (15.2) as a general linear system.

Since the mn eigenvalues of $I_n \otimes A - B^T \otimes I_m$ are given by

$$\lambda_{ij}(I_n \otimes A - B^T \otimes I_m) = \lambda_i(A) - \lambda_j(B), \quad i = 1{:}m, \quad j = 1{:}n, \tag{15.3}$$

the Sylvester equation is nonsingular precisely when A and B have no eigenvalues in common.

In this chapter we briefly discuss the Schur method for solving the Sylvester equation and summarize its rounding error analysis. Then we determine the backward error for the Sylvester equation, investigate its relationship with the residual, and derive a condition number. All these results respect the structure of the Sylvester equation and are relevant to any solution method. We also consider the special case of the Lyapunov equation and mention how the results extend to generalizations of the Sylvester equation.

15.1. Solving the Sylvester Equation

One way to solve the Sylvester equation is to apply Gaussian elimination with partial pivoting (GEPP) to the "big" system (15.2), but the structure of the coefficient matrix cannot be exploited and the cost is a prohibitive $O(m^3 n^3)$ flops. A more efficient method, requiring $O(m^3 + n^3)$ flops, is obtained with the aid of Schur decompositions of A and B. Let A and B have the real Schur decompositions

$$A = URU^T, \qquad B = VSV^T, \tag{15.4}$$

where U and V are orthogonal and R and S are quasi-triangular, that is, block triangular with 1 × 1 or 2 × 2 diagonal blocks, and with any 2 × 2 diagonal blocks having complex conjugate eigenvalues. (See Golub and Van Loan [470, 1989, §7.4.1] for more details of the real Schur decomposition.)

With the decompositions (15.4), the Sylvester equation transforms to

$$RZ - ZS = D \qquad (Z = U^T XV, \quad D = U^T CV), \tag{15.5}$$

or, equivalently, $Pz = d$, where $P = I_n \otimes R - S^T \otimes I_m$, $z = \mathrm{vec}(Z)$ and $d = \mathrm{vet}(D)$. If R and S are both triangular then P is block triangular with triangular diagonal blocks, so $Pz = d$ can be solved by substitution. Expressed in the notation of (15.5), the solution process take the form of n substitutions: if S is upper triangular then

$$(R - s_{jj}I)Z(:,j) = D(:,j) + Z(:,1{:}j-1)S(1{:}j-1,j), \quad j = 1{:}n.$$

Suppose now that R and S are quasi-triangular, and for definiteness assume that they are both upper quasi-triangular. Partitioning $Z = (Z_{ij})$ conformally with $R = (R_{ij})$ and $S = (S_{ij})$ we have

$$R_{ii}Z_{ij} - Z_{ij}S_{jj} = D_{ij} - \sum_{k=i+1}^{m} R_{ik}Z_{kj} + \sum_{k=1}^{j-1} Z_{ik}S_{kj}. \tag{15.6}$$

These equations can be used to determine the blocks of Z working up the block columns from first to last. Since R_{ii} and S_{jj} are both of order 1 or 2, each system (15.6) is a linear system of order 1, 2, or 4 for Z_{ij}; in the latter two cases it is usually solved by GEPP (or even Gaussian elimination with complete pivoting—see Problem 15.4).

This Schur decomposition method for solving the Sylvester equation is due to Bartels and Stewart [74, 1972]. What can be said about its numerical stability? In the case where R and S are both triangular, Theorem 8.5 shows that

$$(P + \Delta P)\widehat{z} = d, \qquad |\Delta P| \le c_{m,n}u|P|, \tag{15.7}$$

where $c_{m,n}$ denotes a constant depending on the dimensions m and n (in fact, we can take $c_{m,n} \approx mn$). Thus $|d - P\widehat{z}| \le c_{m,n}u|P||\widehat{z}|$, which implies the weaker inequality

$$|D - (R\widehat{Z} - \widehat{Z}S)| \le c_{m,n}u(|R||\widehat{Z}| + |\widehat{Z}||S|). \tag{15.8}$$

If R or S is quasi-triangular then the error analysis depends on how the systems of dimension 2 or 4 are solved. If GEPP followed by fixed precision iterative refinement is used for each of these systems "$\overline{P}\overline{z} = \overline{d}$", and if for each system $\overline{P}$ is not too ill conditioned and the vector $|\overline{P}||\overline{z}|$ is not too badly scaled, then (15.7) and (15.8) remain valid (see §11.2). Otherwise, we have only a normwise bound

$$\|D - (R\widehat{Z} - \widehat{Z}S)\|_F \le c'_{m,n}u(\|R\|_F + \|S\|_F)\|\widehat{Z}\|_F.$$

Because the transformation of a matrix to Schur form (by the QR algorithm) is a backward stable process[15] it is true overall that

$$\|C - (A\widehat{X} - \widehat{X}B)\|_F \le c''_{m,n}u(\|A\|_F + \|B\|_F)\|\widehat{X}\|_F. \tag{15.9}$$

Thus the relative residual is guaranteed to be bounded by a modest multiple of the unit roundoff u.

Golub, Nash, and Van Loan [464, 1979] suggested a modification of the Bartels-Stewart algorithm in which A is reduced only to upper Hessenberg form: $A = UHU^T$. The reduced system $HZ - ZS = D$ can be solved by solving n systems that are either upper Hessenberg or differ from upper

---

[15] See Golub and Van Loan [470, 1989, §7.5.6]. A proof is outside the scope of this book, but the necessary tools are Lemmas 18.3 and 18.8 about Householder and Givens rotations.

Hessenberg form by the addition of an extra subdiagonal. As shown in [464, 1979], the Hessenberg–Schur algorithm has a smaller flop count than the Bartels-Stewart algorithm, with the improvement depending on the relative sizes of m and n. The computed solution $\widehat{X}$ again satisfies (15.9).

The use of iterative methods to solve the Sylvester equation has attracted attention recently for applications where A and B are large and sparse [572, 1995], [588, 1992], [935, 1991], [1059, 1988]. The iterations are usually terminated when an inequality of the form (15.9) holds, so here the size of the relative residual is known a priori (assuming the method converges).

15.2. Backward Error

We saw in the last section that standard methods for solving the Sylvester equation are guaranteed to produce a small relative residual. Does a small relative residual imply a small backward error? The answer to this question for a general linear system is yes (Theorem 7. 1). But for the highly structured Sylvester equation the answer must be no, because for the special case of matrix inversion we know that a small residual does not imply a small backward error (§13.1). In this section we investigate the relationship between residual and backward error for the Sylvester equation.

The normwise backward error of an approximate solution Y to (15.1) is defined by

$$\eta(Y) = \min\{\,\epsilon : (A+\Delta A)Y - Y(B+\Delta B) = C + \Delta C,\ \ \|\Delta A\|_F \le \epsilon\alpha,\\ \|\Delta B\|_F \le \epsilon\beta,\ \|\Delta C\|_F \le \epsilon\gamma\,\}. \tag{15.10}$$

The tolerances a, b, and γ provide some freedom in how we measure the perturbations. Of most interest is the choice $a = ||A||_F$, $b = ||B||_F$, $g = ||C||_F$, for which we will call q the *normwise relative backward error*. The equation $(A + DA)Y - Y(B + DB) = C + DC$ may be written

$$\Delta AY - Y\Delta B - \Delta C = R, \tag{15.11}$$

where the residual $R = C - (AY - YB)$. A small backward error implies a small relative residual since, using the optimal perturbations from (15. 10) in (15.11), we have

$$\|R\|_F = \|\Delta AY - Y\Delta B - \Delta C\|_F \le ((\alpha+\beta)\|Y\|_F + \gamma)\eta(Y). \tag{15.12}$$

To explore the converse question of what the residual implies about the backward error we begin by transforming (15.11) using the SVD of Y, $Y = USV^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthogonal and $\Sigma = \mathrm{diag}(s_i)$ 1 $\mathbb{R}^{m\times n}$. The numbers $\sigma_1 \ge \sigma_2 \ge \dots \ge \sigma_{\min(m,n)} \ge 0$ are the singular values of Y and we define, in addition, $\sigma_{\min(m,n)+1} = \dots = \sigma_{\max(m,n)} = 0$. Equation (15.11) transforms to

$$\widetilde{\Delta A}\Sigma - \Sigma\widetilde{\Delta B} - \widetilde{\Delta C} = \widetilde{R}, \tag{15.13}$$

where

$$\widetilde{\Delta A} = U^T \Delta A U, \quad \widetilde{\Delta B} = V^T \Delta B V, \quad \widetilde{\Delta C} = U^T \Delta C V, \quad \widetilde{R} = U^T R V.$$

This is an underdetermined system, with mn equations in $m^2 + n^2 + mn$ unknowns. We can write it in the uncoupled form[16]

$$\frac{\widetilde{\Delta a}_{ij}}{\alpha} \cdot \alpha\sigma_j - \beta\sigma_i \cdot \frac{\widetilde{\Delta b}_{ij}}{\beta} - \gamma \cdot \frac{\widetilde{\Delta c}_{ij}}{\gamma} = \widetilde{r}_{ij}, \quad i = 1\colon m, \quad j = 1\colon n. \tag{15.14}$$

For each i and j it is straightforward to show that the minimum of $(\alpha^{-1}\widetilde{\Delta a}_{ij})^2 + (\beta^{-1}\widetilde{\Delta b}_{ij})^2 + (\gamma^{-1}\widetilde{\Delta c}_{ij})^2$ subject to (15.14) is attained for

$$\begin{aligned}
\frac{\widetilde{\Delta a}_{ij}}{\alpha} &= \frac{\alpha\sigma_j}{\alpha^2\sigma_j^2 + \beta^2\sigma_i^2 + \gamma^2}\widetilde{r}_{ij},\\
\frac{\widetilde{\Delta b}_{ij}}{\beta} &= \frac{-\beta\sigma_i}{\alpha^2\sigma_j^2 + \beta^2\sigma_i^2 + \gamma^2}\widetilde{r}_{ij},\\
\frac{\widetilde{\Delta c}_{ij}}{\gamma} &= \frac{-\gamma}{\alpha^2\sigma_j^2 + \beta^2\sigma_i^2 + \gamma^2}\widetilde{r}_{ij}.
\end{aligned}$$

These matrices minimize

$$\left\| \begin{bmatrix} \dfrac{\widetilde{\Delta A}}{\alpha} & \dfrac{\widetilde{\Delta B}}{\beta} & \dfrac{\widetilde{\Delta C}}{\gamma} \end{bmatrix} \right\|_F \equiv \left(\left\| \frac{\Delta A}{\alpha} \right\|_F^2 + \left\| \frac{\Delta B}{\beta} \right\|_F^2 + \left\| \frac{\Delta C}{\gamma} \right\|_F^2 \right)^{1/2}.$$

Since $h(Y)$ is the minimum value of $\max\{\|a^{-1}DA\|_F, \|b^{-1}DB\|_F, \|\gamma^{-1}DC\|_F\}$, it follows that

$$\frac{\xi}{\sqrt{3}} \le \eta(Y) \le \xi, \tag{15.15}$$

where

$$\begin{aligned}
\xi &:= \min\{\, \| \,[\alpha^{-1}\widetilde{\Delta A} \quad \beta^{-1}\widetilde{\Delta B} \quad \gamma^{-1}\widetilde{\Delta C}]\, \|_F \,\}\\
&= \left(\sum_{i=1}^{m}\sum_{j=1}^{n} \frac{\widetilde{r}_{ij}^{\,2}}{\alpha^2\sigma_j^2 + \beta^2\sigma_i^2 + \gamma^2} \right)^{1/2}.
\end{aligned} \tag{15.16}$$

---

[16] For notational convenience we extend $\widetilde{\Delta A}$ (if $m < n$) or $\widetilde{\Delta B}$ (if $m > n$) to dimension $m \times n$; the "fictitious" elements will be set to zero by the minimization.

This expression shows that the backward error is approximately equal not to the normwise relative residual $\|R\|_F/((\alpha+\beta)\|X\|_F+\gamma)$, but to a componentwise residual corresponding to the diagonalized equation (15.13).

From (15.15) and (15.16) we deduce that

$$\eta(Y) \leq \mu \frac{\|R\|_F}{(\alpha+\beta)\|Y\|_F+\gamma}, \tag{15.17}$$

where

$$\mu := \frac{(\alpha+\beta)\|Y\|_F+\gamma}{(\alpha^2\sigma_n^2+\beta^2\sigma_m^2+\gamma^2)^{1/2}}. \tag{15.18}$$

The scalar $m \geq 1$ is an amplification factor that measures by how much, at worst, the backward error can exceed the normwise relative residual. We now examine m more closely, concentrating on the normwise relative backward error, for which $a = \|A\|_F$, $b = \|B\|_F$, and $\gamma = \|C\|_F$.

First, note that if $n = 1$ and $B = 0$, so that the Sylvester equation reduces to a linear system $Ay = c$, then $\sigma_1 = \|y\|_2$ and $\sigma_k = 0$ for $k > 1$, so $\mu = (\|A\|_F\|y\|_2 + \|c\|_2)/(\|A\|_F^2\|y\|_2^2 + \|c\|_2^2)^{1/2}$. Clearly, $1 \leq \mu \leq \sqrt{2}$, and so we recover Theorem 7.1 (for the 2-norm) from (15.12) and (15.17), to within a constant factor.

If $m = n$ then

$$\mu = \frac{(\|A\|_F+\|B\|_F)\|Y\|_F+\|C\|_F}{((\|A\|_F^2+\|B\|_F^2)\sigma_{\min}(Y)^2+\|C\|_F^2)^{1/2}}. \tag{15.19}$$

We see that m is large only when

$$\|Y\|_F \gg \sigma_{\min}(Y) \quad \text{and} \quad \|Y\|_F \gg \frac{\|C\|_F}{\|A\|_F+\|B\|_F}, \tag{15.20}$$

that is, when Y is ill conditioned and Y is a large-normed solution to the Sylvester equation. In the general case, with $m \neq n$, one of σ_m^2 and σ_n^2 is always zero and hence m can be large for a third reason: A (if $m < n$) or B (if $m > n$) greatly exceeds the rest of the data in norm; in these cases the Sylvester equation is badly scaled. However, if we set $a = b = \|A\|_F + \|B\|_F$, which corresponds to regarding A and B as comprising a single set of data, then bad scaling does not affect m.

If we allow only A and B to be perturbed in (15.10) (as may be desirable if the right-hand side C is known exactly), then $\gamma = 0$ and (15.19) and (15.20) remain valid with $\|C\|_F$ replaced by zero. In this case $m \geq \|Y\|_F\|Y^+\|_2 \approx \kappa_2(Y)$ (for any m and n), so m is large whenever Y is ill conditioned (and included in this case is matrix inversion). Conditions involving controllability which guarantee that the solution to the Sylvester equation with $m = n$ is nonsingular are given by Hearon [508, 1977], while Datta [265, 1988] gives a determinantal condition for nonsingularity. It is an open problem to derive conditions for the Sylvester equation to have a well-conditioned solution (see Problem 15.5).

The following numerical example illustrates the above analysis. This particular example was carefully chosen so that the entries of A and B are of a simple form, but equally effective examples are easily generated using random, ill-conditioned A and B of dimension m, $n > 2$. Let

$$A = \begin{bmatrix} 1 & -1 \\ 1 & -1 \end{bmatrix}, \quad B = A - \alpha \begin{bmatrix} 1+\alpha & 0 \\ 0 & 1 \end{bmatrix}.$$

Define C by the property that vet(C) is the singular vector corresponding to the smallest singular value of $I_n \otimes A - B^T \otimes I_m$. With $a = 10^{-6}$, we solved the Sylvester equation in MATLAB by the Bartels-Stewart algorithm and found that the computed solution X satisfies

$$\frac{\|R\|_F}{(\|A\|_F + \|B\|_F)\|\widehat{X}\|_F + \|C\|_F} = 2.82 \times 10^{-17}, \quad \sigma(\widehat{X}) = \{2 \times 10^{18}, 5 \times 10^5\},$$

$$\eta(\widehat{X}) \approx \xi = 2.21 \times 10^{-8}, \quad \mu = 5.66 \times 10^{12}.$$

Although $\widehat{X}$ has a very acceptable residual (as it must in view of (15.9)), its backward error is eight orders of magnitude larger than is necessary to achieve backward stability. We solved the same Sylvester equation using GEPP on the system (15.2). The relative residual was again less than u, but the backward error was appreciably larger: $\eta(\widehat{X}) \approx 1.53 \times 10^{-5}$.

One conclusion we can draw from the analysis is that *standard methods for solving the Sylvester equation are at best conditionally backward stable*, since there exist rounding errors such that $\widetilde{r}_{m,n}$ is the only nonzero element of $\widetilde{R}$, and then (15.17) is an approximate equality, with m possibly large.

15.2.1. The Lyapunov Equation

If we put $B = -A^T$ in the Sylvester equation we obtain

$$AX + XA^T = C,$$

which is called the *Lyapunov equation*. This equation plays a major role in control and systems theory and it can be solved using the same techniques as for the Sylvester equation.

If $C = C^T$ then $C = AX + XA^T = X^TA^T + AX^T = C^T$, so X and X^T are both solutions to the Lyapunov equation. If the Lyapunov equation is nonsingular (equivalently, $\lambda_i(A) + \lambda_j(A) \neq$ for all i and j, by (15.3)) it therefore has a unique symmetric solution.

We assume that C is symmetric and that Y is a symmetric approximate solution. The definition of backward error is now

$$\eta(Y) = \min\{\,\epsilon : (A+\Delta A)Y + Y(A+\Delta A)^T = C + \Delta C,\ \ \|\Delta A\|_F \le \epsilon\alpha, \\ \Delta C = \Delta C^T,\ \ \|\Delta C\|_F \le \epsilon\gamma\,\}.$$

The analogue of (15.11) is $AY + Y\Delta A^T - \Delta C = R := C - AY - YA^T$. Let $Y = ULU^T$ be a spectral decomposition, with $\Lambda = \mathrm{diag}(\lambda_i)$. Then the residual equation transforms to

$$\widetilde{\Delta A}\Lambda + \Lambda\widetilde{\Delta A}^T - \widetilde{\Delta C} = \widetilde{R},$$

where $\widetilde{\Delta A} = U^T\Delta AU$, $\widetilde{\Delta C} = U^T\Delta CU$, and $\widetilde{R} = U^TRU$. This system can be written in uncoupled form as

$$\frac{\widetilde{\Delta a}_{ij}}{\alpha}\cdot\alpha\lambda_j + \alpha\lambda_i\cdot\frac{\widetilde{\Delta a}_{ji}}{\alpha} - \gamma\cdot\frac{\widetilde{\Delta c}_{ij}}{\gamma} = \widetilde{r}_{ij}, \qquad i,j = 1\!:\!n. \tag{15.21}$$

We can obtain the minimum value of $\|\,[\alpha^{-1}\Delta A \quad \gamma^{-1}\Delta C]\,\|_F$ by minimizing

$$(\alpha^{-1}\widetilde{\Delta a}_{ij})^2 + (\alpha^{-1}\widetilde{\Delta a}_{ji})^2 + 2(\gamma^{-1}\widetilde{\Delta c}_{ij})^2$$

subject to (15.21), for i, j = 1:n. The solution is

$$\frac{\widetilde{\Delta a}_{ij}}{\alpha} = \frac{2\alpha\lambda_j}{2\alpha^2(\lambda_i^2+\lambda_j^2)+\gamma^2}\widetilde{r}_{ij}, \qquad \frac{\widetilde{\Delta c}_{ij}}{\gamma} = \frac{-\gamma}{2\alpha^2(\lambda_i^2+\lambda_j^2)+\gamma^2}\widetilde{r}_{ij}.$$

(Note that $\widetilde{\Delta C}$ is symmetric since $\widetilde{R}$ is.) It follows that $\xi/\sqrt{2} \le \eta(Y) \le \xi$, where

$$\xi^2 = \sum_{i,j=1}^{n}\frac{(4\alpha^2\lambda_j^2+\gamma^2)\widetilde{r}_{ij}^{\,2}}{(2\alpha^2(\lambda_i^2+\lambda_j^2)+\gamma^2)^2} \le \sum_{i,j=1}^{n}\frac{2\widetilde{r}_{ij}^{\,2}}{(2\alpha^2(\lambda_i^2+\lambda_j^2)+\gamma^2)},$$

where the last inequality is usually a good approximation. Comparing with (15.16) we see that respecting the extra structure of the Lyapunov equation has essentially no effect on the backward error.

Finally, the analogue of (15.17) and (15.18) is

$$\eta(Y) \le \mu\frac{\|R\|_F}{2\alpha\|Y\|_F+\gamma}, \qquad \mu := \frac{\sqrt{2}(2\alpha\|Y\|_F+\gamma)}{(4\alpha^2\lambda_*^2+\gamma^2)^{1/2}},$$

where $\lambda_* = \min_i|\lambda_i|$.

15.3. Perturbation Result

To derive a perturbation result we consider the perturbed Sylvester equation

$$(A+\Delta A)(X+\Delta X)-(X+\Delta X)(B+\Delta B)=C+\Delta C,$$

which, on dropping second-order terms, becomes

$$A\Delta X-\Delta XB=\Delta C-\Delta AX+X\Delta B.$$

This system may be written in the form

$$P\,\mathrm{vec}(\Delta X)=-\left[\,X^T\otimes I_m \quad -I_n\otimes X \quad -I_{mn}\,\right]\begin{bmatrix}\mathrm{vec}(\Delta A)\\ \mathrm{vec}(\Delta B)\\ \mathrm{vec}(\Delta C)\end{bmatrix}, \qquad (15.22)$$

where $P = I_n \otimes A - B^T \otimes I_m$. If we measure the perturbations normwise by

$$\epsilon=\max\left\{\frac{\|\Delta A\|_F}{\alpha},\frac{\|\Delta B\|_F}{\beta},\frac{\|\Delta C\|_F}{\gamma}\right\},$$

where a, b, and γ are tolerances as in (15.10), then

$$\frac{\|\Delta X\|_F}{\|X\|_F}\le\sqrt{3}\Psi\epsilon \qquad (15.23)$$

is a sharp bound (to first order in ϵ), where

$$\Psi=\|P^{-1}\left[\alpha(X^T\otimes I_m)\quad -\beta(I_n\otimes X)\quad -\gamma I_{mn}\right]\|_2/\|X\|_F \qquad (15.24)$$

is the corresponding condition number for the Sylvester equation. The bound (15.23) can be weakened to

$$\frac{\|\Delta X\|_F}{\|X\|_F}\le\sqrt{3}\Phi\epsilon, \qquad (15.25)$$

where

$$\Phi=\|P^{-1}\|_2\frac{(\alpha+\beta)\|X\|_F+\gamma}{\|X\|_F}.$$

If $\|P^{-1}\|_2(\alpha+\beta)\epsilon<1/2$ then twice the upper bound in (15.25) can be shown to be a strict bound for the error. The perturbation bound (15.25) with a $\|A\|_F$, b = $\|B\|_F$, and $\gamma = \|C\|_F$ is the one that is usually quoted in the literature for the Sylvester equation (see [464, 1979] and [522, 1988], for example), and corresponds to applying standard perturbation theory for $Ax = b$ to (15.2). Note that $\|P^{-1}\|_2 = \mathrm{sep}(A, B)^{-1}$, where sep is the separation of A and B,

$$\mathrm{sep}(A,B)=\min_{X\neq 0}\frac{\|AX-XB\|_F}{\|X\|_F}. \qquad (15.26)$$

The sep function is an important tool for measuring invariant subspace sensitivity [470, 1989, §7.2.5], [940, 1973], [1050, 1979].

For the Lyapunov equation, a similar derivation to the one above shows that the condition number is

$$\|(I_n \otimes A + A \otimes I_n)^{-1}\,[\,\alpha\big((X^T \otimes I_n) + (I_n \otimes X)\Pi^T\big), \quad -\gamma I_{n^2}\,]\,\|_2/\|X\|_F, \tag{15.27}$$

where P is the vet-permutation matrix, which is defined by the property that $\mathrm{vec}(A^T) = P\ \mathrm{vet}(A)$.

How much can the bounds (15.23) and (15.25) differ? The answer is by an arbitrary factor. To show this we consider the case where B is normal (or equivalently, A is normal if we transpose the Sylvester equation). We can assume B is in Schur form, thus $B = \mathrm{diag}(m_j)$ (with the m_j possibly complex). Then $P = \mathrm{diag}(A - u_{jj}I_m)$, and it is straightforward to show that if $X = [x_1, \ldots, x_n]$, and if we approximate the 2-norms in the definitions of Ψ and Φ by Frobenius norms, then

$$\Psi^2 \approx \Bigg(\alpha^2 \sum_{j=1}^{n} \|x_j\|_2^2 \|(A - \mu_{jj}I_m)^{-1}\|_F^2 + \beta^2 \sum_{j=1}^{n} \|(A - \mu_{jj}I_m)^{-1}X\|_F^2$$
$$+\gamma^2 \sum_{j=1}^{n} \|(A - \mu_{jj}I_m)^{-1}\|_F^2\Bigg)\Big/\|X\|_F^2,$$

while

$$\Phi^2 \approx \sum_{j=1}^{n} \|(A - \mu_{jj}I_m)^{-1}\|_F^2\big((\alpha+\beta) + \gamma/\|X\|_F\big)^2.$$

These formulae show that in general Ψ and Φ will be of similar magnitude, and we know that $\Psi \le \Phi$ from the definitions. However, Ψ can be much smaller than Φ. For example, suppose that $\gamma = 0$ and

$$\|(A - \mu_{nn}I_m)^{-1}\|_F \gg \max_{j \ne n} \|(A - \mu_{jj}I_m)^{-1}\|_F.$$

Then if

$$\frac{\|x_n\|_2}{\|X\|_F} \ll 1 \quad \text{and} \quad \frac{\|(A - \mu_{nn}I_m)^{-1}X\|_F}{\|X\|_F} \ll \|(A - \mu_{nn}I_m)^{-1}\|_F,$$

we have $\Psi << \Phi$. Such examples are easily constructed. To illustrate, let $A = \mathrm{diag}(2, 2, \ldots, 2, 1)$ and $B = \mathrm{diag}(1/2, 1/2, \ldots, 1/2, 1 - \epsilon)$, with $\epsilon > 0$, so that $A - m_{nn}I_m = \mathrm{diag}(1+\epsilon, 1+\epsilon, \ldots, 1+\epsilon, \epsilon)$, and let $X = (A - m_{nn}I_m)Y$, where $Y = [y, y, \ldots, y, 0]$ with $\|(A - m_{nn}I_m)y\|_2 = \|A - m_{nn}I_m\|_2$ and $\|y\|_2 = 1$. Then, if $\gamma = O(\epsilon)$

$$\Psi = O(\alpha^2 + \beta^2), \quad \Phi \approx \epsilon^{-1}(\alpha^2 + \beta^2).$$

To summarize, the "traditional" perturbation bound (15.25) for the Sylvester equation can severely overestimate the effect of a perturbation on the data when only A and B are perturbed, because it does not take account of the special structure of the problem. In contrast, the perturbation bound (15.23) does respect the Kronecker structure, and consequently is attainable for any given A, B, and C.

To obtain an a posteriori error bound for a computed solution $\widehat{X} := X + DX$ we can set $DA = 0$, $DB = 0$, and $DC = AX - XB - C = R$ in (15.22), which leads to

$$\frac{\|X-\widehat{X}\|_F}{\|X\|_F} \le \|P^{-1}\|_2 \frac{\|R\|_F}{\|X\|_F}. \tag{15.28}$$

A similar but potentially much smaller bound is described in the next section.

15.4. Practical Error Bounds

For the Sylvester equation we can obtain an analogue of the practical error bound (7.27) by identifying $Ax = b$ with (15.2). For the computed residual of a computed solution X we have

$$\widehat{R} = fl(C-(fl(A\widehat{X}) - fl(\widehat{X}B))) = R + \Delta R,$$
$$\Delta R| \le u|C| + \gamma_{m+2}|A||\widehat{X}| + \gamma_{n+2}|\widehat{X}||B| =: R_u.$$

Therefore the bound is

$$\frac{\|X-\widehat{X}\|}{\|\widehat{X}\|} \le \frac{\|\,|P^{-1}|(|\operatorname{vec}(\widehat{R})| + \operatorname{vec}(R_u))\,\|}{\|\widehat{X}\|}, \tag{15.29}$$

where $\|X\| := \max_{i,j} |x_{ij}|$. After transformation by the technique illustrated in (14.1), this bound can be estimated by the LAPACK norm estimator (Algorithm 14.4) at the cost of solving a few linear systems with coefficient matrices $I_n \otimes A - B^T \otimes I_m$ and its transpose—in other words, solving a few Sylvester equations $AX - XB = C$ and $A^TX - XB^T = D$. If the Bartels-Stewart algorithm is used, these solutions can be computed with the aid of the previously computed Schur decompositions of A and B. The condition number Ψ in (15.24) and $\operatorname{sep}(A, B) = \|P^{-1}\|_2^{-1}$ can both be estimated in much the same way; alternatively, the power method can be used (see Ghavimi and Laub [440, 1995]). Other algorithms for efficiently estimating sep(A, B) given Schur decompositions of A and B are given by Byers [172, 1984] and Kågström and Poromaa [621, 1992].

The attraction of (15.29) is that large elements in the jth column of P^{-1} may be countered by a small jth element of $\operatorname{vec}(\widehat{R}) + \operatorname{vec}(R_u)$, making the bound much smaller than (15.28). In this sense (15.29) has better scaling properties than (15.28), although (15.29) is not actually invariant under diagonal scalings of the Sylvester equation.

We give a numerical example to illustrate the advantage of (15.29) over (15.28). Let

$$A = J_3(0), \quad B = J_3(10^{-3}), \quad c_{ij} \equiv 1,$$

where $J_n(\lambda)$ denotes a Jordan block of size n with eigenvalue λ. Solving the Sylvester equation by the Bartels–Stewart algorithm we found that the bounds are

$$(15.28) : 8.00 \times 10^{-3}, \qquad (15.29) : 6.36 \times 10^{-15}$$

(where in evaluating (15.28) we replaced R by $|\widehat{R}| + R_u$, as in (15.29)). Here, $\operatorname{sep}(A, B) = 1.67 \times 10^{-16}$, and the bound (15.29) is small because relatively large columns of P^{-1} are nullified by relatively small elements of $|\operatorname{vec}(\widehat{R})| + \operatorname{vec}(R_u)$. For this example, with $a = \|A\|_F$, $b = \|B\|_F$, $\gamma = \|C\|_F$, we have

$$\Psi = 7.00 \times 10^9, \qquad \Phi = 1.70 \times 10^{16},$$

confirming that the usual perturbation bound (15.25) for the Sylvester equation can be very pessimistic. Furthermore,

$$\frac{\|R\|_F}{(\|A\|_F + \|B\|_F)\|\widehat{X}\|_F + \|C\|_F} = 7.02 \times 10^{-24},$$

$$\sigma(\widehat{X}) = \{6 \times 10^{15},\ 5 \times 10^{8},\ 3 \times 10^{2}\},$$

$$\eta(\widehat{X}) \approx \xi = 1.00 \times 10^{-19}, \quad \mu = 2.26 \times 10^{13},$$

so we have an example where the backward error is small despite a large m.

15.5. Extensions

The Sylvester equation can be generalized in two main ways. One retains the linearity but adds extra coefficient matrices, yielding the *generalized Sylvester equations*

$$AXB + CXD = E \tag{15.30}$$

and

$$AX - YB = C, \quad DX - YE = F. \tag{15.31}$$

These two forms are equivalent, under conditions on the coefficient matrices [210, 1987]; for example, defining $Z := XB$ and $W := -CX$, (15.30) becomes $AZ - WD = E$, $CZ + WB = 0$. Applications of generalized Sylvester equations include the computation of stable eigendecompositions of matrix pencils [294, 1987], [295, 1988], [622, 1993], [623, 1994] and the implementation of numerical methods for solving implicit ordinary differential equations [353, 1980].

The second generalization incorporates a quadratic term, yielding the *algebraic Riccati equation*

$$AX + XB - XFX + G = 0. \tag{15.32}$$

This general Riccati equation and its symmetric counterpart with $B = AT$ and F and G symmetric are widely used in control theory.

The backward error results and perturbation theory of this chapter can be generalized in a straightforward way to (15.31) and (15.32). See Kågström [620, 1994] for (15.31) and Ghavimi and Laub [440, 1995] for (15.32). The backward error derivations do not extend to (15.30), because in this equation the coefficient matrices appear nonlinearly.

A variation of the Lyapunov equation called the *discrete-time Lyapunov equation* has the form

$$X - F^TXF = Q,$$

where $F, Q \in \mathbb{R}^{n\times n}$. As in (15.30), the data appears nonlinearly. Ghavimi and Laub [441, 1995] show how to derive an approximation to the backward error by linearizing an equation characterizing the optimal perturbations.

Another generalization of the Sylvester equation, mainly of theoretical interest, is

$$\sum_{i=1}^{k} A_iXB_i = C,$$

where $A_i \in \mathbb{R}^{m\times m}$ and $B_i \in \mathbb{R}^{n\times n}$, $i = 1\colon k$. See Lancaster [685, 1970] for associated theory.

15.6. Notes and References

This chapter is based on Higham [556, 1993]. The backward error derivations make use of ideas of Ghavimi and Laub [440, 1995].

The Sylvester equation is so named because Sylvester considered the homogeneous version of the equation [985, 1884].

Bhatia and Rosenthal [96, 1996] give a survey of theoretical results for the Sylvester equation in both finite- and infinite-dimensional spaces.

For details of the role of the Sylvester equation in the eigenproblem see Bai, Demmel, and McKenney [38, 1993], [40, 1993] and the references therein.

Iterative methods that make use of matrix inversion to solve the Sylvester equation are described by Miller [755, 1988] and Roberts [875, 1980].

Hammarling [496, 1982] gives a method for solving the Lyapunov equation $AX + XA^T = -C$ in the case where A has eigenvalues with negative real parts and C is positive semidefinite; his method directly computes the Cholesky factor of the solution (which is indeed symmetric positive definite—see Problem 15.2).

A survey of the vec operator, the Kronecker product, and the vec-permutation matrix is given together with historical comments by Henderson and Searle [514, 1981]. Historical research by Henderson, Pukelsheim, and Searle [513, 1983] indicates that the Kronecker product should be called the Zehfuss product, in recognition of an 1858 paper by Zehfuss that gives a determinantal result involving the product.

The vet-permutation matrix P (which appears in (15.27)) is given explicitly by

$$\Pi = \sum_{i,j=1}^{n} (e_i e_j^T) \otimes (e_j e_i^T),$$

and has the property that $(A \otimes B)\Pi = \Pi(B \otimes A)$.

Applications of the Lyapunov equation in control theory, including special situations where an approximate solution of low rank is required, are discussed by Hodel [574, 1992]. A much older reference to applications is Barnett and Storey [68, 1968].

Algorithms and software for solving (15.30) are developed by Gardiner, Wette, Laub, Amato, and Moler [417, 1982], [418, 1982].

Perturbation theory for Lyapunov and Riccati equations can be found in the work of Byers [173, 1985], Hewer and Kenney [522, 1988], [650, 1990], and Gahinet, Laub, Kenney, and Hewer [411, 1990].

Chu [210, 1987] determines conditions for the existence of unique solutions to the generalized Sylvester equations (15.30) and (15.31). The appropriate conditions for (15.30) are that the pencils $A + \lambda C$ and $D + \lambda B$ are regular and the spectra of the pencils have an empty intersection, which neatly generalizes the conditions for the Sylvester equation to have a unique solution; the conditions for (15.31) are analogous.

There is much work on algorithms and software for solving the algebraic Riccati equation. For a sampling, see Laub [691, 1979], Arnold and Laub [30, 1984], Byers [174, 1987], Gardiner, and Laub [416, 1991], and Kenney, Laub, and Papadopoulos [653, 1992].

An algorithm for estimating a generalization of sep that occurs in perturbation theory for the generalized Sylvester equation (15.31) is developed by Kågström and Westin [624, 1989].

Another generalization of the Sylvester equation is to take just one equation from (15.31), $AX - YB = C$ ((15.13) is of this form). This equation can be underdetermined or overdetermined, depending on the dimensions of the coefficient matrices. Conditions involving generalized inverses that are both necessary and sufficient for the existence of a solution are given by Baksalary and Kala [49, 1979]. Zietak examines the inconsistent case [1130, 1985] for one choice of dimensions giving an overdetermined system. Stewart [949, 1992] shows how to compute a minimum Frobenius norm least squares solution. The even more general equation $AXB + CYD = E$ has also been analysed by Baksalary and Kala [50, 1980], who again give necessary and sufficient conditions for the existence of a solution.

15.6.1. LAPACK

The computations discussed in this chapter can all be done using LAPACK. The Bartels-Stewart algorithm can be implemented by calling `xGEES` to compute the Schur decomposition, using the level-3 BLAS routine `xGEMM` to transform the right-hand side C, calling `xTRSYL` to solve the (quasi-) triangular Sylvester equation, and using `xGEMM` to transform back to the solution X. The error bound (15.29) can be estimated using `xLACON` in conjunction with the above routines. A Fortran 77 code `dggsvx` [556, 1993] of Higham follows this outline and may appear in a future release of LAPACK.

Routine `xLASY2` solves a real Sylvester equation $AX \pm XB = sC$ in which A and B have dimension 1 or 2 and σ is a scale factor. It is called by `xTRSYL`.

Kågström and Poromaa have developed codes for solving (15.31), which are intended for a future release of LAPACK [622, 1993], [623, 1994].

Problems

15.1. Show that the Sylvester equation $AX - XA = I$ has no solution.

15.2. (Bellman [89, 1970, §10.18]) Show that if the expression

$$X = -\int_0^\infty e^{At} C e^{Bt} \, dt$$

exists for all C it represents the unique solution of the Sylvester equation $AX + XB = C$. (Hint: consider the matrix differential equation $dZ/dt = AZ(t) + Z(t)B$, $Z(0) = C$.) Deduce that the Lyapunov equation $AX + XA^T = -C$ has a symmetric positive definite solution if A has eigenvalues with negative real parts and C is symmetric positive definite.

15.3. (Byers and Nash [176, 1987]) Let $A \in \mathbb{R}^{n \times n}$ and consider

$$\mathrm{sep}(A, -A^T) = \min_{X \neq 0} \frac{\|AX + XA^T\|_F}{\|X\|_F}.$$

Show that there exists a minimizer X that is either symmetric or skew-symmetric.

15.4. How would you solve a Sylvester equation $AX - XB = C$ in which A and B are of dimension 1 or 2? Compare your method with the one used in the LAPACK routine `XLASY2`.

15.5. (RESEARCH PROBLEM) Derive conditions for the Sylvester equation to have a well-conditioned solution.

Chapter 16

Stationary Iterative Methods

I recommend this method to you for imitation.
You will hardly ever again eliminate directly,
at least not when you have more than 2 unknowns.
The indirect [iterative] procedure can be done while half asleep,
or while thinking about other things.[17]
— CARL FRIEDRICH GAUSS, *Letter to C. L. Gerling (1823)*

The iterative method is commonly called the "Seidel process, "
or the "Gauss–Seidel process. "
But, as *Ostrowski* (1952) *points out,*
Seidel (1874) *mentions the process but advocates* not *using it.*
Gauss nowhere mentions it.
— GEORGE E. FORSYTHE,
Solving Linear Algebraic Equations Can Be Interesting (1953)

The spurious contributions in null(A)
grow at worst linearly and
if the rounding errors are small the scheme can be quite effective.
— HERBERT B. KELLER,
On the Solution of Singular and Semidefinite
Linear Systems by Iteration (1965)

---

[17] Gauss refers here to his relaxation method for solving the normal equations. The translation is taken from Forsythe [387, 1951].

Table 16.1. *Dates of publication of selected iterative methods. Based on Young* [1123, 1989].

1845	Jacobi	Jacobi method
1874	Seidel	Gauss-Seidel method
1910	Richardson	Richardson's method
1938–1939	Temple	Method of steepest descent
1940s	Various (analysis by Young and Frankel)	Successive overrelaxation (SOR) method
1952	Hestenesand Stiefel	Conjugate gradient method

Iterative methods for solving linear systems have along history, going back at least to Gauss. Table 16.1 shows the dates of publication of selected methods. It is perhaps surprising, then, that rounding error analysis for iterative methods is not well developed. There are two main reasons for the paucity of error analysis. One is that in many applications accuracy requirements are modest and are satisfied without difficulty, resulting in little demand for error analysis. Certainly there is no point in computing an answer to greater accuracy than that determined by the data, and in scientific and engineering applications the data often has only a few correct digits. The second reason is that rounding error analysis for iterative methods is inherently more difficult than for direct methods, and the bounds that are obtained are harder to interpret.

In this chapter we consider a simple but important class of iterative methods, stationary iterative methods, for which a reasonably comprehensive error analysis can be given. The basic question that our analysis attempts to answer is "What is the limiting accuracy of a method in floating point arithmetic?" Specifically, how small can we guarantee that the backward or forward error will be over all iterations $k = 1, 2, \ldots$? Without an answer to this question we cannot be sure that a convergence test of the form $\|b - A\widehat{x}_k\| \le \epsilon$ (say) will ever be satisfied, for any given value of $\epsilon < \|b - Ax_0\|$!

As an indication of the potentially devastating effects of rounding errors we present an example constructed and discussed by Hammarling and Wilkinson [497, 1976]. Here, A is the 100×100 lower bidiagonal matrix with $a_{ii} \equiv 1.5$ and $a_{i,i-1} \equiv 1$, and $b_i \equiv 2.5$. The successive overrelaxation (SOR) method is applied in MATLAB with parameter $w = 1.5$, starting with the rounded version of the exact solution x, given by $x_i = 1 - (-2/3)^i$. The forward errors $\|\widehat{x}_k - x\|_\infty / \|x\|_\infty$ and the co-norm backward errors $\eta_{A,b}(\widehat{x}_k)$ are plotted in Figure 16.1. The SOR method converges in exact arithmetic, since the iteration matrix has spectral radius 1/2, but in the presence of rounding errors it diverges. The iterate $\widehat{x}_{238}$ has a largest element of order 10^{13}, $\widehat{x}_{k+2} \equiv \widehat{x}_k$ for

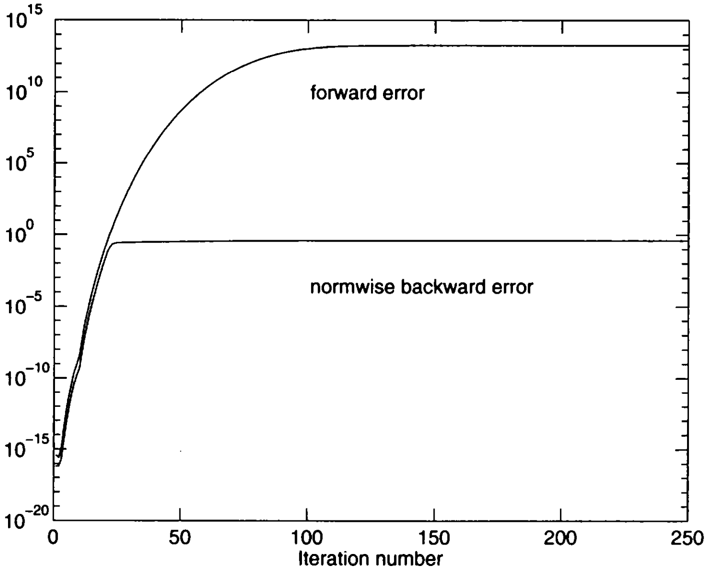


Figure 16.1. *SOR iteration.*

$k \geq 238$, and for $k > 100$, $\widehat{x}_k(60\colon 100) \approx (-1)^k \widehat{x}_{100}(60\colon 100)$. The divergence is not a result of ill conditioning of A, since $\kappa_\infty(A) \approx 5$. 'The reason for the initial rapid growth of the errors in this example is that the iteration matrix is far from normal; this allows the norms of its powers to become very large before they ultimately decay by a factor $\approx 1/2$ with each successive power. The effect of rounding errors is to cause the forward error curve in Figure 16.1 to level off near $k = 100$, instead of decaying to zero as it would in exact arithmetic. More insight into the initial behaviour of the errors can be obtained using the notion of pseudo-eigenvalues; see §17.3.

16.1. Survey of Error Analysis

Before analysing stationary iterative methods, we briefly survey the published error analysis for iterative methods. For symmetric positive definite systems, Golub [468, 1962] derives both statistical and nonstatistical bounds for the forward error and residual of the Richardson method. Benschop and Ratz [92, 1971] give a statistical analysis of the effect of rounding errors on stationary iteration, under the assumption that the rounding errors are independent random variables with zero mean. Lynn [719, 1964] presents a statistical analysis for the SOR method with a symmetric positive definite matrix.

Hammarling and Wilkinson [497, 1976] give a normwise error analysis for the SOR method. With the aid of numerical examples, they emphasize that while it is the spectral radius of the iteration matrix $M^{-1}N$ that determines the asymptotic rate of convergence, it is the norms of the powers of this matrix that govern the behaviour of the iteration in the early stages. This point is also explained by Trefethen [1017, 1992], using the tool of pseudospectra.

Dennis and Walker [302, 1984] obtain bounds for $\|x - \widehat{x}_{k+1}\|/\|x - \widehat{x}_k\|$ for stationary iteration as a special case of error analysis of quasi-Newton methods for nonlinear systems. The bounds in [302, 1984] do not readily yield information about normwise or componentwise forward stability.

Bollen [133, 1984] analyses the class of "descent methods" for solving $Ax = b$, where A is required to be symmetric positive definite; these are obtained by iteratively using exact line searches to minimize the quadratic function $F(x) = (A^{-1}b - x)^T A(A^{-1}b - x)$. The choice of search direction $p_k = b - Ax_k =: r_k$ yields the steepest descent method, while $p_k = e_j$ (unit vector), where $|r_k|_j = \|r_k\|_\infty$, gives the Gauss-Southwell method. Bollen shows that both methods are normwise backward stable as long as a condition of the form $c_n\kappa(A)u < 1$ holds. If the p_k are cyclically chosen to be the unit vectors $e_1, e_2, \ldots, e_n$ then the Gauss–Seidel method results, but unfortunately no results specific to this method are given in [133, 1984].

Wozniakowski [1112, 1977] shows that the Chebyshev semi-iterative method is normwise forward stable but not normwise backward stable, and in [1113, 1978] he gives a normwise error analysis of stationary iterative methods. Some of the assumptions in [1113, 1978] are difficult to justify, as explained by Higham and Knight [563, 1993].

In [1114, 1980] Wozniakowski analyses a class of conjugate gradient algorithms (which does not include the usual conjugate gradient method). He obtains a forward error bound proportional to $\kappa(A)^{3/2}$ and a residual bound proportional to $K(A)$, from which neither backward nor forward normwise stability can be deduced. We note that as part of the analysis in [1114, 1980] Wozniakowski obtains a residual bound for the steepest descent method that is proportional to $K(A)$, and is therefore much weaker than the bound obtained by Bollen [133, 1984].

Zawilski [1125, 1991] shows that the cyclic Richardson method for symmetric positive definite systems is normwise forward stable provided the parameters are suitably ordered. He also derives a sharp bound for the residual that includes a factor $\kappa(A)$, and which therefore shows that the method is not normwise backward stable.

Arioli and Romani [29, 1992] give a statistical error analysis of stationary iterative methods. They investigate the relations between a statistically defined asymptotic stability factor, ill conditioning of $M^{-1}A$, where $A = M - N$ is the splitting, and the rate of convergence.

Greenbaum [479, 1989] presents a detailed error analysis of the conjugate gradient method, but her concern is with the rate of convergence rather than the attainable accuracy. An excellent survey of work concerned with the effects of rounding error on the conjugate gradient method (and the Lanczos method) is given by Greenbaum and Strakos in the introduction of [480, 1992]; see also Greenbaum [481, 1994]. Notay [799, 1993] analyses how rounding errors influence the convergence rate of the conjugate gradient method for matrices with isolated eigenvalues at the ends of the spectrum. Van der Vorst [1058, 1990] examines the effect of rounding errors on preconditioned conjugate gradient methods with incomplete Cholesky preconditioners.

The analysis given in the remainder of this chapter is from Higham and Knight [563, 1993], [564, 1993], wherein more details are given. Error analysis of Kaczmarz's row-action method is given by Knight [663, 1993].

16.2. Forward Error Analysis

A stationary iterative method has the form

$$Mx_{k+1} = Nx_k + b,$$

where $A = M - N \in \mathbb{R}^{n\times n}$ is nonsingular and M is nonsingular. We assume that the spectral radius $\rho(M^{-1}N) < 1$, so that in exact arithmetic the iteration converges for any starting vector x_0. We are not concerned with the size of constants in this analysis, so we denote by c_n a constant of order n.

The computed vectors $\widehat{x}_k$ satisfy an equality of the form

$$(M + \Delta M_{k+1})\widehat{x}_{k+1} = N\widehat{x}_k + b + f_k,$$

which we write as

$$M\widehat{x}_{k+1} = N\widehat{x}_k + b - \xi_k, \tag{16.1}$$

where

$$\xi_k = \Delta M_{k+1}\widehat{x}_{k+1} - f_k.$$

We will assume that M is triangular (as is the case for the Jacobi, Gauss–Seidel, SOR, and Richardson iterations), so that $|\Delta M_{k+1}| \le c_n' u|M|$ and f_k accounts solely for the errors in forming $N\widehat{x}_k + b$. Hence

$$|\xi_k| \le c_n u(|M||\widehat{x}_{k+1}| + |N||\widehat{x}_k| + |b|) =: \mu_k. \tag{16.2}$$

Solving the recurrence (16.1) we obtain

$$\widehat{x}_{m+1} = G^{m+1}x_0 + \sum_{k=0}^{m} G^k M^{-1}(b - \xi_{m-k}), \tag{16.3}$$

where $G = M^{-1}N$. Since the iteration is stationary at x,

$$x = G^{m+1}x + \sum_{k=0}^{m} G^k M^{-1}b, \tag{16.4}$$

and so the error $e_{m+1} := x - \widehat{x}_{m+1}$ satisfies

$$e_{m+1} = G^{m+1}e_0 + \sum_{k=0}^{m} G^k M^{-1} \xi_{m-k}. \tag{16.5}$$

We have

$$|e_{m+1}| \le |G^{m+1}e_0| + \sum_{k=0}^{m} |G^k M^{-1}| \, \mu_{m-k}, \tag{16.6}$$

where m_k is the bound for x_k defined in (16.2). The first term, $|G^{m+1}e_0|$, is the error of the iteration in exact arithmetic and is negligible for large m. The accuracy that can be guaranteed by the analysis is therefore determined by the last term in (16.6), and it is this term on which the rest of the analysis focuses.

At this point we can proceed by using further componentwise inequalities or by using norms. First we consider the norm approach. By taking norms in (16.6) and defining

$$\gamma_x = \sup_k \frac{\|\widehat{x}_k\|_\infty}{\|x\|_\infty}, \tag{16.7}$$

we obtain

$$\begin{aligned} \|e_{m+1}\|_\infty &\le \|G^{m+1}e_0\|_\infty + \max_{0\le k\le m} \|\mu_k\|_\infty \sum_{k=0}^{m} \|G^k M^{-1}\|_\infty \\ &\le \|G^{m+1}e_0\|_\infty + c_n u(1+\gamma_x)(\|M\|_\infty + \|N\|_\infty)\|x\|_\infty \\ &\qquad \times \sum_{k=0}^{\infty} \|G^k M^{-1}\|_\infty, \end{aligned} \tag{16.8}$$

where the existence of the sum is assured by the result of Problem 16.1.

If $\|G\|_\infty = \|M^{-1}N\|_\infty = q < 1$ then (16.8) yields

$$\|e_{m+1}\|_\infty \le \|G^{m+1}e_0\|_\infty + c_n u(1+\gamma_x)(\|M\|_\infty + \|N\|_\infty)\|x\|_\infty \frac{\|M^{-1}\|_\infty}{1-q}.$$

Thus if q is not too close to 1 ($q \le 0.9$, say), and g_x and $\|M^{-1}\|_\infty$ are not too large, a small forward error is guaranteed for sufficiently large m.

Of more interest is the following componentwise development of (16.6). Defining

$$\theta_x = \sup_k \max_{1\le i\le n} \left(\frac{|\widehat{x}_k|_i}{|x_i|} \right), \tag{16.9}$$

so that $|\widehat{x}_k| \le \theta_x |x|$ for all k, we have from (16.2),

$$|\mu_k| \le c_n u(1+\theta_x)(|M| + |N|)|x|. \tag{16.10}$$

Hence (16.6) yields

$$|e_{m+1}| \le |G^{m+1}e_0| + c_n u(1+\theta_x)\left(\sum_{k=0}^{\infty}|G^k M^{-1}|\right)(|M|+|N|)|x|, \qquad (16.11)$$

where, again, the existence of the sum is assured by the result of Problem 16.1. Since $A = M - N = M(I - M^{-1}N)$ we have

$$A^{-1} = \left(\sum_{k=0}^{\infty}(M^{-1}N)^k\right)M^{-1}.$$

The sum in (16.11) is clearly an upper bound for $|A^{-1}|$. Defining $c(A) \ge 1$ by

$$c(A) = \min\left\{\epsilon : \sum_{k=0}^{\infty}|(M^{-1}N)^k M^{-1}| \le \epsilon\left|\sum_{k=0}^{\infty}(M^{-1}N)^k M^{-1}\right| = \epsilon\,|A^{-1}|\right\} \qquad (16.12)$$

we have our final bound

$$|e_{m+1}| \le |G^{m+1}e_0| + c_n u(1+\theta_x)c(A)|A^{-1}|(|M|+|N|)|x|. \qquad (16.13)$$

An interesting feature of stationary iteration methods is that if the elements of M and N are multiples of the elements in the corresponding positions of A, then any scaling of the form $Ax = b \rightarrow D_1AD_2 \cdot D_2^{-1}x = D_1b$ (D_i diagonal) leaves the eigenvalues of $M^{-1}N$ unchanged; hence the asymptotic convergence rate is independent of row and column scaling. This scale independence applies to the Jacobi and SOR iterations, but not, for example, to the stationary Richardson iteration, for which $M = I$. One of the benefits of doing a componentwise analysis is that under the above assumptions on M and N the bound (16.13) largely shares the scale independence. In (16.13) the scalar $c(A)$ is independent of the row and column scaling of A, and the term $|A^{-1}|(|M| + |N|)|x|$ scales in the same way as x. Furthermore, q_x can be expected to depend only mildly on the row and column scaling, because the bound in (16.2) for the rounding error terms has the correct scaling properties.

What can be said about $c(A)$? In general, it can be arbitrarily large. Indeed, $c(A)$ is infinite for the Jacobi and Gauss–Seidel iterations for any $n \ge 3$ if A is the symmetric positive definite matrix with $a_{ij} = \min(i, j)$, because A^{-1} is tridiagonal and $(M^{-1}N)^k M^{-1}$ is not.

If M^{-1} and $M^{-1}N$ both have nonnegative elements then $c(A) = 1$; as we will see in the next section, this condition holds in some important instances.

Some further insight into $c(A)$ can be obtained by examining the case where $M^{-1}N \in \mathbb{C}^{n\times n}$ is diagonal with eigenvalues λ_i It is easy to show that $c(A) = \max_i |1-\lambda_i|/(1-|\lambda_i|)$, so $c(A)$ *can* be large only if $\rho(M^{-1}N)$ is close to 1. Although $M^{-1}N$ cannot be diagonal for the Jacobi or Gauss–Seidel methods, this formula can be taken as being indicative of the size of $c(A)$ when $M^{-1}N$ is diagonalizable with a well-conditioned matrix of eigenvectors. We therefore have the heuristic inequality, for general A,

$$c(A) \geq \max_i \frac{|1-\lambda_i|}{1-|\lambda_i|}, \qquad \lambda_i = \lambda_i(M^{-1}N). \tag{16.14}$$

In practical problems where stationary iteration is used, we would expect $c(A)$ to be of modest size ($O(n)$, say) for two reasons. First, to achieve a reasonable convergence rate, $\rho(M^{-1}N)$ has to be safely less than 1, which implies that the heuristic lower bound (16.14) for $c(A)$ is not too large. Second, even if A is sparse, A^{-1} will usually be full, and so there are unlikely to be zeros on the right-hand side of (16.12). (Such zeros are dangerous because they can make $c(A)$ infinite.)

Note that in (16.13) the only terms that depend on the history of the iteration are $|G^{m+1}e_0|$ and q_x. In using this bound we can redefine x_0. to be any iterate $\widehat{x}_k$ thereby possibly reducing q_x. This is a circular argument if used to obtain a priori bounds, but it does suggest that the potentially large q_x term will generally be innocuous. Note that if $x_i = 0$ for some i then q_x is infinite unless $(\widehat{x}_k)_i = 0$ for all k. This difficulty with zero components of x can usually be overcome by redefining

$$\theta_x = \sup_k \max_{1\leq i\leq n} \frac{\big((|M|+|N|)|\widehat{x}_k|\big)_i}{\big((|M|+|N|)|x|\big)_i},$$

for which the above bounds remain valid if q_x is replaced by $2q_x$.

Finally, we note that (16.13) implies

$$\|e_{m+1}\|_\infty \leq \|G^{m+1}e_0\|_\infty + c_n u(1+\theta_x)c(A)\|\,|A^{-1}|(|M|+|N|)|x|\,\|_\infty. \tag{16.15}$$

If $q_x c(A) = O(1)$ and $|M| + |N| \leq a|A|$, with $a = O(1)$, this bound is of the form $c_n \operatorname{cond}(A,x)u$ as $m \to \infty$, and we have componentwise forward stability.

Now we specialize the forward error bound (16.15) to the Jacobi, Gauss–Seidel, and SOR iterations.

16.2.1. Jacobi's Method

For the Jacobi iteration, $M = D = \operatorname{diag}(A)$ and $N = \operatorname{diag}(A) - A$. Hence $|M| + |N| = |M - N| = |A|$, and so (16.15) yields

$$\|e_{m+1}\|_\infty \leq \|G^{m+1}e_0\|_\infty + c_n u(1+\theta_x)c(A)\|\,|A^{-1}||A||x|\,\|_\infty. \tag{16.16}$$

If A is an M-matrix then $M^{-1} \geq 0$ and $M^{-1} \geq 0$, so $c(A) = 1$. Hence in this case we have componentwise forward stability as $m \to \infty$ if q_x is suitably bounded.

Table 16.2. *Jacobi method,* $a = 1/2 - 8^{-j}$.

	$\rho(M^{-1}N)$	Iters.	cond(A,x)	$\min_k \phi(\widehat{x}_k)$	$\min_k \eta_{A,b}(\widehat{x}_k)$
$j = 1$	0.75	90	3.40	2.22e-16	1.27e-16
$j = 2$	0.97	352	4.76	1.78-15	9.02e-16
$j = 3$	0.996	1974	4.97	1.42e-14	7.12e-15
$j = 4$	1.00	11226	5.00	1.14e-13	5.69e-14
$j = 5$	1.00	55412	5.00	9.10e-13	4.55e-13

Table 16.3. *Jacobi method,* $a = -(1/2-8^{-j})$.

	$\rho(M^{-1}N)$	Iters.	cond(A,x)	$\min_k \phi(\widehat{x}_k)$	$\min_k \eta_{A,b}(\widehat{x}_k)$
$j = 1$	0.75	39	7.00	4.44e-16	5.55e-17
$j = 2$	0.97	273	6.30e1	4.88e-15	7.63e-17
$j = 3$	0.996	1662	5.11e2	4.22e-14	8.24e-17
$j = 4$	1.00	9051	4.09e3	3.41e-13	8.32e-17
$j = 5$	1.00	38294	3.28e4	2.73e-12	8.33e-17

Wozniakowski [1113, 1978, Ex. 4.1] cites the symmetric positive definite matrix

$$A = \begin{bmatrix} 1 & a & a \\ a & 1 & a \\ a & a & 1 \end{bmatrix}, \quad 0 < a < \frac{1}{2}, \quad \kappa_2(A) = \frac{1+2a}{1-a}, \quad \rho(M^{-1}N) = 2a,$$

as a matrix for which the Jacobi method can be unstable, in the sense that there exist rounding errors such that no iterate has a relative error bounded by $c_n \kappa_\infty(A) u$. Let us see what our analysis predicts for this example. Straightforward manipulation shows that if $a = 1/2 - \epsilon$ $(\epsilon > 0)$, then $c(A) \approx (3\epsilon)^{-1}$, so $c(A) \to \infty$ as $\epsilon \to 0$. (The heuristic lower bound (16.14) is approximately $3(2\epsilon)^{-1}$ in this case.) Therefore (16.16) suggests that the Jacobi iteration can be unstable for this matrix. To confirm the instability we applied the Jacobi method to the problem with $x = [1,\ 1,\ 1]^T$ and $a = 1/2 - 8^{-j}$, $j = 1{:}5$. We took a random x_0. with $\|x - x_0\|_2 = 10^{-10}$, and the iteration was terminated when there was no decrease in the norm of the residual for 50 consecutive iterations. Table 16.2 reports the smallest value of $\phi(\widehat{x}_k) = \|x - \widehat{x}_k\|_\infty / \|x\|_\infty$ over all iterations, for each j; the number of iterations is shown in the column "Iters."

The ratio $\min_k \phi(\widehat{x}_k)_{j+1} / \min_k \phi(\widehat{x}_k)_j$ takes the values 8.02, 7.98, 8.02, 7.98 for j = 1:4, showing excellent agreement with the behaviour predicted by (16.16), since $c(A) \approx 8^j/3$. Moreover, $\theta_x \approx 1$ in these tests and setting $c_n \approx 1$ the bound (16.16) is at most a factor 13.3 larger than the observed error, for each *j*.

If $-1/2 < a < 0$ then *A* is an *M*-matrix and $c(A) = 1$. The bound (16.16) shows that if we set $a = -(1/2 - 8^{-j})$ and repeat the above experiment then the Jacobi method will perform in a componentwise forward stable manner (clearly, $\theta_x \approx 1$ is to be expected). We carried out the modified experiment, obtaining the results shown in Table 16.3. All the $\min_k \phi(\widehat{x}_k)_j$ values are less than $\mathrm{cond}(A,x)u$, so the Jacobi iteration is indeed componentwise forward stable in this case. Note that since $\rho(M^{-1}N)$ and $\|M^{-1}N\|_2$ take the same values for a and $-a$, the usual rate of convergence measures cannot distinguish between these two examples.

16.2.2. Successive Overrelaxation

The SOR method can be written in the form $Mx_{k+1} = Nx_{k+b}$, where

$$M = \frac{1}{\omega}(D + \omega L), \quad N = \frac{1}{\omega}((1-\omega)D - \omega U),$$

and where $A = D + L + U$, with *L* and *U* strictly lower triangular and upper triangular, respectively. The matrix $|M| + |N|$ agrees with $|A|$ everywhere except, possibly, on the diagonal, and the best possible componentwise inequality between these two matrices is

$$|M| + |N| \le \frac{1 + |1-\omega|}{\omega}|A| \equiv f(\omega)|A|. \tag{16.17}$$

Note that $f(w) = 1$ for $1 \le w \le 2$, and $f(\omega) \to \infty$ as $\omega \to 0$. From (16.15) we have

$$\|e_{m+1}\|_\infty \le \|G^{m+1}e_0\|_\infty + c_n u(1+\theta_x)c(A)f(\omega)\|\,|A^{-1}||A||x|\,\|_\infty.$$

If *A* is an Al-matrix and $0 \le w \le 1$ then $M^{-1} \ge 0$ and $M^{-1}N \ge 0$, so $c(A) = 1$. The Gauss–Seidel method corresponds to $w = 1$, and it is interesting to note that for this method the forward error bound has exactly the same form as that for the Jacobi method (though $c(A)$ and q_x are, of course, different for the two methods).

16.3. Backward Error Analysis

We now turn our attention to bounding the residual vector, $r_k = b - A\widehat{x}_k$.

From (16.3) and (16.4) we find that

$$r_{m+1} = AG^{m+1}(x - x_0) + \sum_{k=0}^{m} AG^k M^{-1}\xi_{m-k}.$$

It is easy to show that $AG^k = H^kA$, where $H := NM^{-1}$ (recall that $G = M^{-1}N$). Therefore

$$r_{m+1} = H^{m+1}r_0 + \sum_{k=0}^{m} H^k(I - H)\xi_{m-k}. \tag{16.18}$$

Taking norms and using (16.2) gives, similarly to (16.8),

$$\|r_{m+1}\|_\infty \le \|H^{m+1}r_0\|_\infty + c_n u\sigma(1+\gamma_x)(\|M\|_\infty + \|N\|_\infty)\|x\|_\infty, \tag{16.19}$$

where

$$\sigma = \left\|\sum_{k=0}^{\infty} |H^k(I - H)|\right\|_\infty.$$

The following bound shows that s is small if $\|H\|_\infty = q < 1$, with q not too close to 1:

$$\sigma \le \|I - H\|_\infty \sum_{k=0}^{\infty} \|H\|_\infty^k = \frac{\|I - H\|_\infty}{1 - q}.$$

A potentially much smaller bound can be obtained under the assumption that H is diagonalizable. If $H = XDX^{-1}$, with $D = \mathrm{diag}(\lambda_i)$, then

$$\begin{aligned}
\sigma &= \left\|\sum_{k=0}^{\infty} |X(D^k - D^{k+1})X^{-1}|\right\|_\infty \\
&\le \left\| |X| \left(\sum_{k=0}^{\infty} \mathrm{diag}(|1 - \lambda_i||\lambda_i^k|)\right)|X^{-1}| \right\|_\infty \\
&= \left\| |X|\,\mathrm{diag}\left(\frac{|1 - \lambda_i|}{1 - |\lambda_i|}\right)|X^{-1}| \right\|_\infty \\
&\le \kappa_\infty(X) \max_i \frac{|1 - \lambda_i|}{1 - |\lambda_i|}.
\end{aligned} \tag{16.20}$$

Note that $\lambda_i = \lambda_i(H) = \lambda_i(NM^{-1}) = \lambda_i(M^{-1}N)$, so we see the reappearance of the term in the heuristic bound (16.14). The bound (16.20) is of modest size if the eigenproblem for H is well conditioned ($\kappa_\infty(X)$ is small) and $p(H)$ is not too close to 1. Note that real eigenvalues of H near +1 do not affect the bound for s, even though they may cause slow convergence.

To summarize, (16.19) shows that, for large m, the normwise backward error $\eta_{A,b}(\widehat{x}_m)$ for the ∞-norm is certainly no larger than

$$c_n u(1+\gamma_x)\left(\frac{\|M\|_\infty + \|N\|_\infty}{\|A\|_\infty}\right)\sigma.$$

Note that $\|M\|_\infty + \|N\|_\infty \le 2\|A\|_\infty$ for the Jacobi and Gauss-Seidel methods, and also for the SOR method if $w \ge 1$.

A componentwise residual bound can also be obtained, but it does not lead to any identifiable classes of matrix or iteration for which the componentwise relative backward error is small.

To conclude, we return to our numerical examples. For the SOR example at the start of the chapter, $c(A) = O(10^{45})$ and $s = O(10^{30})$, so our error bounds for this problem are all extremely large. In this problem $\max_i |1 - \lambda_i|/(1 - |\lambda_i|) = 3$, where $\lambda_i = \lambda_i(M^{-1}N)$, so (16.14) is very weak; (16.20) is not applicable since $M^{-1}N$ is defective.

For the first numerical example in §16.2.1, Table 16.2 reports the minimum ∞-norm backward errors $\eta_{A,b}(\widehat{x}_k)$. For this problem it is straightforward to show that $\sigma = (1 - \epsilon)/\epsilon = 8^j(1 - 8^{-j})$. The ratios of backward errors for successive value of j are 7.10, 7.89, 7.99, 8.00, so we see excellent agreement with the behaviour predicted by the bounds. Table 16.3 reports the normwise backward errors for the second numerical example in §16.2.1. The backward errors are all less than u, which again is close to what the bounds predict, since it can be shown that $s \le 5$ for $-1/2 \le a \le 0$. In both of the examples of §16.2.1 the componentwise backward error $\omega_{|A|,|b|}(\widehat{x}_k) \approx \eta_{A,b}(\widehat{x}_k)$, and in our practical experience this behaviour is typical or the Jacobi and SOR iterations.

16.4. Singular Systems

Singular linear systems occur in a variety of applications, including the computation of the stationary distribution vector in a Markov chain [94, 1994], [647, 1983] and the solution of a Neumann boundary value problem by finite difference methods [834, 1976]. Because of the structure and the possibly large dimension of the coefficient matrices in these applications, iterative solution methods are frequently used. An important question is how the rather delicate convergence properties of the iterative methods are affected by rounding errors. In this section we extend the analysis of stationary iteration to singular systems.

16.4.1. Theoretical Background

A useful tool in analysing the behaviour of stationary iteration for a singular system is the *Drazin inverse*. This can be defined, for $A \in \mathbb{R}^{n\times n}$, as the unique matrix A^D such that

$$A^DAA^D = A^D, \quad AA^D = A^DA, \quad \text{and} \quad A^{k+1}A^D = A^k,$$

where $k = \operatorname{index}(A)$. The index of A is the smallest nonnegative integer k such that $\operatorname{rank}(A^k) = \operatorname{rank}(A^{k+1})$; it is characterized as the dimension of the largest Jordan block of A with eigenvalue zero. If $\operatorname{index}(A) = 1$ then A^D

is also known as the *group inverse* of A and is denoted by $A^{\#}$. The Drazin inverse is an "equation-solving inverse" precisely when index$(A) \le 1$, for then $AA^DA = A$, and so if $Ax = b$ is a consistent system then A^Db is a solution. As we will see, however, the Drazin inverse of the coefficient matrix A itself plays no role in the analysis. The Drazin inverse can be represented explicitly as follows. If

$$A = P \begin{bmatrix} B & 0 \\ 0 & N \end{bmatrix} P^{-1},$$

where P and B are nonsingular and N has only zero eigenvalues, then

$$A^D = P \begin{bmatrix} B^{-1} & 0 \\ 0 & 0 \end{bmatrix} P^{-1}.$$

Further details of the Drazin inverse can be found in Campbell and Meyer's excellent treatise [180, 1979, Chap. 7].

Let $A \in \mathbb{R}^{n \times n}$ be a singular matrix and consider solving $Ax = b$ by stationary iteration with a splitting $A = M - N$, where M is nonsingular. First, we examine the convergence of the iteration in exact arithmetic. Since any limit point x of the sequence $\{x_k\}$ must satisfy $Mx = Nx + b$, or $Ax = b$, *we* restrict our attention to consistent linear systems. (For a thorough analysis of stationary iteration for inconsistent systems see Dax [269, 1990].) As in the nonsingular case we have the relation (cf. (16.4)):

$$x_{m+1} = G^{m+1}x_0 + \sum_{i=0}^{m} G^i M^{-1} b, \tag{16.21}$$

where $G = M^{-1}N$. Since A is singular, G has an eigenvalue 1, so G^m does not tend to zero as $m \to \infty$, that is, G is not *convergent.* If the iteration is to converge for all x_0 then $\lim_{m\to\infty} G^m$ must exist. Following Meyer and Plemmons [752, 1977], we call a matrix B for which $\lim_{m\to\infty} B^m$ exists *semiconvergent.*

We assume from this point on that G is semiconvergent. It is easy to see [94, 1994, Lem. 6.9] that G must have the form

$$G = P \begin{bmatrix} I & 0 \\ 0 & \Gamma \end{bmatrix} P^{-1}, \tag{16.22}$$

where P is nonsingular and $\rho(\Gamma) < 1$. Hence

$$\lim_{m\to\infty} G^m = P \begin{bmatrix} I & 0 \\ 0 & 0 \end{bmatrix} P^{-1}.$$

To rewrite this limit in terms of G, we note that

$$I - G = P \begin{bmatrix} 0 & 0 \\ 0 & I - \Gamma \end{bmatrix} P^{-1}, \tag{16.23}$$

and, since $I - \Gamma$ is nonsingular,

$$(I-G)^D = P\begin{bmatrix} 0 & 0 \\ 0 & (I-\Gamma)^{-1} \end{bmatrix} P^{-1}. \tag{16.24}$$

Hence

$$\lim_{m\to\infty} G^m = I - (I-G)^D(I-G). \tag{16.25}$$

To evaluate the limit of the second term in (16.21) we note that, since the system is consistent, $M^{-1}b = M^{-1}Ax = (I-G)x$, and so

$$\begin{aligned}\sum_{i=0}^{m} G^i M^{-1} b &= \sum_{i=0}^{m} G^i (I-G)x \\ &= (I - G^{m+1})x \\ &\to (I-G)^D(I-G)x = (I-G)^D M^{-1} b.\end{aligned}$$

We note in passing that the condition that G is semiconvergent is equivalent to $I - G$ having index 1, in view of (16.23), but that this condition does not imply that $A = M(I-G)$ has index 1.

The conclusion is that if G is semiconvergent, stationary iteration converges to a solution of $AX = b$ that depends on x_0:

$$\lim_{m\to\infty} x_m = (I - (I-G)^D(I-G))x_0 + (I-G)^D M^{-1} b. \tag{16.26}$$

The first term in this limit is in null$(I - G)$ and the second term is in range$(I - G)$. To obtain the unique solution in range$(I - G)$ we should take for x_0. any vector in range$(I - G)$ ($x_0 = 0$, say).

16.4.2. Forward Error Analysis

We wish to bound $e_{m+1} = x - \widehat{x}_{m+1}$, where x is the limit in (16.26) corresponding to the given starting vector x_0. The analysis proceeds as in the nonsingular case, up to the derivation of equation (16.5):

$$e_{m+1} = G^{m+1}e_0 + \sum_{i=0}^{m} G^i M^{-1} \xi_{m-i}.$$

As before, the first term, $G^{m+1}e_0$, is negligible for large m, because it is the error after $m + 1$ stages of the exact iteration and this error tends to zero. To obtain a useful bound for the second term, we cannot simply take norms or absolute values, because $\sum_{i=0}^{m} G^i$ grows unfoundedly with m (recall that G has an eigenvalue 1). Our approach is to split the vectors ξ_i according to $\xi_i = \xi_i^{(1)} + \xi_i^{(2)}$, where $M^{-1}\xi_i^{(1)} \in \mathrm{range}(I-G)$ and $M^{-1}\xi_i^{(2)} \in$

null$(I - G)$; this is a well-defined splitting because range$(I - G)$ and null$(I - G)$ are complementary subspaces (since index$(I - G) = 1$, or equivalently, G is semiconvergent). Using the properties of the splitting the error can be written as

$$\begin{aligned} e_{m+1} &= G^{m+1}e_0 + \sum_{i=0}^{m} G^i M^{-1}\xi_{m-i}^{(1)} + \sum_{i=0}^{m} G^i M^{-1}\xi_{m-i}^{(2)} \\ &= G^{m+1}e_0 + \sum_{i=0}^{m} G^i M^{-1}\xi_{m-i}^{(1)} + M^{-1}\sum_{i=0}^{m} \xi_{m-i}^{(2)}. \end{aligned}$$

We achieve the required splitting for x_i via the formulae

$$\xi_i^{(1)} = MEM^{-1}\xi_i, \quad \xi_i^{(2)} = M(I-E)M^{-1}\xi_i,$$

where

$$E = (I-G)^D(I-G).$$

Hence the error can be written as

$$e_{m+1} = G^{m+1}e_0 + \sum_{i=0}^{m} G^i EM^{-1}\xi_{m-i} + (I-E)M^{-1}\sum_{i=0}^{m}\xi_{m-i}. \tag{16.27}$$

Clearly, as $m \to \infty$ the final term in this expression can become unbounded, but since it grows only linearly in the number of iterations it is unlikely to have a significant effect in applications where stationary iteration converges quickly enough to be of practical use.

Now we bound the term

$$S_m = \sum_{i=0}^{m} G^i EM^{-1}\xi_{m-i}. \tag{16.28}$$

Using inequality (16.2) and the definition of g_x in (16.7) and q_x in (16.9), we have

$$\|S_m\|_\infty \le c_n u(1+\gamma_x)(\|M\|_\infty + \|N\|_\infty)\|x\|_\infty \sum_{i=0}^{m} \|G^i EM^{-1}\|_\infty,$$

$$|S_m| \le c_n u(1+\theta_x)\sum_{i=0}^{m} |G^i EM^{-1}|(|M|+|N|)|x|. \tag{16.29}$$

The convergence of the two infinite sums is assured by the result of Problem 16.1, since by (16.22)-(16.24),

$$\begin{aligned} G^iE &= G^i(I-G)^D(I-G) \\ &= P\begin{bmatrix} I & 0 \\ 0 & \Gamma^i \end{bmatrix}P^{-1}\cdot P\begin{bmatrix} 0 & 0 \\ 0 & (I-\Gamma)^{-1} \end{bmatrix}P^{-1}\cdot P\begin{bmatrix} 0 & 0 \\ 0 & (I-\Gamma) \end{bmatrix}P^{-1} \\ &= P\begin{bmatrix} 0 & 0 \\ 0 & \Gamma^i \end{bmatrix}P^{-1} = (GE)^i \quad (i \ge 1), \end{aligned} \tag{16.30}$$

where $\rho(\Gamma) < 1$.

We conclude that we have the normwise error bound

$$\begin{aligned}\|e_{m+1}\|_\infty \le \|G^{m+1}e_0\|_\infty + c_n u(1+\gamma_x)(\|M\|_\infty + \|N\|_\infty)\|x\|_\infty \\ \times \Big\{\sum_{i=0}^{\infty}\|G^i E M^{-1}\|_\infty + (m+1)\|(I-E)M^{-1}\|_\infty\Big\}. \quad (16.31)\end{aligned}$$

On setting $E = I$ we recover the result (16.8) for the nonsingular case. If we assume that G is diagonal, so that P in (16.30) is a matrix of eigenvectors of G, then

$$\sum_{i=0}^{\infty}\|G^i E M^{-1}\|_\infty \le \kappa_\infty(P)\|M^{-1}\|_\infty \frac{1}{1-\rho(\Gamma)}.$$

This bound shows that a small forward error is guaranteed if $\kappa(P)\|M^{-1}\| = O(1)$ and the second largest eigenvalue of G is not too close to 1. (It is this subdominant eigenvalue that determines the asymptotic rate of convergence of the iteration.)

Turning to the componentwise case, we see from (16.24) and (16.30) that

$$\sum_{i=0}^{\infty} G^i E = (I-G)^D.$$

Because of the form of the sum in (16.29), this prompts us to define the scalar $c(A) \ge 1$ by

$$c(A) = \min\Big\{\epsilon : \sum_{i=0}^{\infty}|G^i E M^{-1}| \le \epsilon|(I-G)^D M^{-1}|\Big\},$$

in terms of which we have the componentwise error bound

$$\begin{aligned}|e_{m+1}| \le |G^{m+1}e_0| + c_n u(1+\theta_x)\big\{c(A)|(I-G)^D M^{-1}| \\ + (m+1)|(I-E)M^{-1}|\big\}(|M|+|N|)|x|. \quad (16.32)\end{aligned}$$

Again, as a special case we have the result for nonsingular A, (16.13).

To what should we compare this bound? A perturbation result for $Ax = b$ is given in [564, 1993] that projects the perturbations of A and b into range$(I - G)$ and thus can be thought of as gauging the effect of perturbations to the "nonsingular part of the system". For perturbations of order ϵ it gives an expression

$$\Delta x = (I-G)^D M^{-1}(\Delta b - \Delta A x) + O(\epsilon^2).$$

Hence we can deduce conditions on a stationary iterative method that ensure it is componentwise forward stable, in the sense of yielding a solution whose error is no larger than the uncertainty in x caused by rounding the data. The constants q_x and $c(A)$ should be bounded by d_n, where d_n denotes a slowly growing function of n; the inequality $|M| + |N| \le d'_n|A|$ should hold, as it does for the Jacobi method and for the SOR method when $\omega \in [\beta, 2]$, where b is positive and not too close to zero; and the "exact error" $G^{m+1}e_0$ must decay quickly enough to ensure that the term $(m + 1)\,|\,(I - E)M^{-1}|$ does not grow too large before the iteration is terminated.

Numerical results given in [564, 1993] show that the analysis can correctly predict forward and backward stability, and that for certain problems linear growth of the component of the error in null(A) can indeed cause an otherwise convergent iteration to diverge, even when starting very close to a solution.

16.5. Stopping an Iterative Method

What convergence test should be used to stop an iterative linear equation solver? In this section we explain how backward errors and condition numbers help answer this question. Note first that most iterative methods for solving $Ax = b$ compute all or part of a matrix–vector product $w = Au$ on each iteration, and in floating point arithmetic we have

$$\widehat{w} = (A + \Delta A)v, \qquad |\Delta A| \le \gamma_m|A|,$$

where m is the maximum number of nonzeros per row of A. The method therefore cannot distinguish between A and $A + DA$ where $|DA| \le g_m\,|A|$, and so there is no point in trying to achieve a componentwise relative backward error less than g_m. Of course, instability of a method (or simply lack of convergence) may pose further restrictions on how small a backward error can be achieved.

It is worth keeping in mind throughout this discussion that in practical applications accuracy and stability requirements are often quite modest because of large errors or uncertainties in the data, or because the iteration is an "inner step" of some "outer" iterative process. Indeed, one of the advantages of iterative methods is that the slacker the convergence tolerance the less computational effort is required, though the relation between tolerance and work depends very much on the method.

Natural stopping criteria for an iterative method are that some measure of backward error or forward error does not exceed a tolerance, ϵ. We will assume that the residual $r = b - Ay$ is available for each iterate y, and that norms of y, r, and A can be computed or estimated. If r is not computed directly, but is recurred by the method, as, for example, in the conjugate gradient method, then the norm of the computed residual may differ from that of the true residual by several orders of magnitude; clearly, this affects the way that the stopping tests are interpreted.

From Theorem 7.1 we have the following equivalences, for any subordinate matrix norm:

$$\|r\| \le \epsilon\|b\| \iff Ay = b + \Delta b, \quad \|\Delta b\| \le \epsilon\|b\|, \tag{16.33a}$$

$$\|r\| \le \epsilon\|A\|\,\|y\| \iff (A + \Delta A)y = b, \quad \|\Delta A\| \le \epsilon\|A\|, \tag{16.33b}$$

$$\begin{aligned} \|r\| \le \epsilon\big(\|A\|\,\|y\| + \|b\|\big) \iff & (A + \Delta A)y = b + \Delta b, \\ & \|\Delta A\| \le \epsilon\|A\|, \quad \|\Delta b\| \le \epsilon\|b\|. \end{aligned} \tag{16.33c}$$

These inequalities remain true with norms replaced by absolute values (Theorem 7.3), but to evaluate (16.33b) and (16.33c) a matrix–vector product $|A||y|$ must be computed, which is a nontrivial expense in an iterative method.

Of these tests, (16.33c) is preferred in general, assuming it is acceptable to perturb both A and b. Note the importance of including both $||A||$ and $||y||$ in the test on $||r||$; a test $||r|| \le \epsilon\|A\|$, though scale independent, does not bound any relative backward error. Test (16.33a) is commonly used in existing codes, but may be very stringent, and possibly unsatisfiable. To see why, note that the residual of the rounded exact solution $fl(x) = x + Dx$, $|Dx| \le u|x|$, satisfies, for any absolute norm,

$$\|r\| = \|b - A(x + \Delta x)\| \le u\|A\|\,\|x\|,$$

and

$$\frac{\|A\|\,\|x\|}{\kappa(A)} \le \|b\| \le \|A\|\,\|x\|.$$

If A is ill conditioned and x is a large-normed solution (that is, $||x|| \approx ||A^{-1}||||b||$), so that $||b||$ is close to its lower bound, then (16.33a) is much harder to satisfy than (16.33c).

If the forward error is to be bounded, then, for a nonsingular problem, tests can be derived involving the residual and A^{-1}: the equality $x - y = A^{-1}r$ leads to normwise and componentwise forward error bounds, such as $||x - y||/||y|| \le ||A^{-1}||||r||/||y||$. Since these bounds involve A^{-1}) they are nontrivial to compute. Some iterative methods automatically produce estimates of the extremal eigenvalues, and hence of $\kappa_2(A) = \lambda_{\max}(A)/\lambda_{\min}(A)$. For large, sparse symmetric positive definite matrices $||A^{-1}||_2$ can be cheaply estimated using the Lanczos method. Another possibility is to use condition estimation techniques (Chapter 14).

The discussion in this section has inevitably been very general. Other considerations in practice include detecting nonconvergence of a method (due to rounding errors or otherwise), adapting the tests to accommodate a preconditioned (the residual r provided by the method may now be that for the preconditioned system), and using a norm that corresponds to a quantity being minimized by the method (a norm that may be nontrivial to compute).

16.6. Notes and References

The Gauss–Seidel method was chosen by Wilkinson [1080, 1948] as an "example of coding" for the ACE. Speaking of experience at that time at the National Physical Laboratory, he explained that "In general, direct methods have been used on those equations which did not yield readily to relaxation, and hence those solved by direct methods have nearly always been of an ill-conditioned type".

Stationary iterative methods are relatively easy to program, although there are many issues to consider when complicated data structures or parallel machines are used. A good source of straightforward C, Fortran, and MATLAB implementations of the Jacobi, Gauss–Seidel, and SOR methods, and other nonstationary iterative methods, is the book *Templates for the Solution of Linear Systems* [70, 1994]; the software is available from netlib. The book contains theoretical discussions of the methods, together with practical advice on topics such as data structures and the choice of a stopping criterion. The choice of stopping criteria for iterative methods is also discussed by Arioli, Duff, and Ruiz [28, 1992].

An up-to-date textbook on iterative methods is Axelsson [34, 1994].

Problems

16.1. Show that if $B \in \mathbb{R}^{n\times n}$ and $\rho(B) < 1$, then the series $\sum_{k=0}^{\infty} |B^k|$ and $\sum_{k=0}^{\infty} \|B^k\|$ are both convergent, where $\|\cdot\|$ is any norm.

16.2. (Descloux [304, 1963]) Consider the (nonlinear) iterative process

$$x_{k+1} = G(x_k) + e_k, \qquad x_0 \text{ given},$$

where $G : \mathbb{R}^n \to \mathbb{R}^n$ satisfies

$$\|G(x) - a\| \le \beta\|x - a\| \qquad \text{for all } x \in \mathbb{R}^n, \tag{16.34}$$

for some $\beta \in (0,1)$, and where $\|e_k\| \le \alpha$ for all k. Note that a must satisfy $G(a) = a$.

(a) Show that

$$\begin{aligned} &\text{if } \|x_k - a\| \le \frac{\alpha}{1-\beta} \quad \text{then} \quad \|x_{k+1} - a\| \le \frac{\alpha}{1-\beta};\\ &\text{if } \|x_k - a\| > \frac{\alpha}{1-\beta} \quad \text{then} \quad \|x_{k+1} - a\| < \|x_k - a\|. \end{aligned}$$

(b) Show that the sequence $\{x_k\}$ is bounded and its points of accumulation x satisfy

$$\|x - a\| \le \frac{\alpha}{1-\beta}.$$

(c) Explain the practical relevance of the result of (b).

Chapter 17
Matrix Powers

Unfortunately, the roundoff errors in the mth power of a matrix, say B^m*,*
are usually small relative to $||B||^m$ *rather than* $||B^m||$*.*
— CLEVE B. MOLER and CHARLES F. VAN LOAN,
Nineteen Dubious Ways to Compute the Exponential of a Matrix (1978)

It is the size of the hump that matters:
the behavior of $||p(ADt)^n|| = ||p(ADt)^{t/D t}||$ *for small but nonzero t.*
The eigenvalues and the norm, by contrast, give sharp information
only about the limits $t \to \infty$ *or* $t \to 0$.
— DESMOND J. HIGHAM and LLOYD N. TREFETHEN,
Stiffness of ODES (1993)

Powers of matrices occur in many areas of numerical analysis. One approach to proving convergence of multistep methods for solving differential equations is to show that a certain parameter-dependent matrix is uniformly "power bounded" [493, 1991, §V.7], [862, 1992]. Stationary iterative methods for solving linear equations converge precisely when the powers of the iteration matrix converge to zero. And the power method for computing the largest eigenvalue of a matrix computes the action of powers of the matrix on a vector. It is therefore important to understand the behaviour of matrix powers, in both exact and finite precision arithmetic.

It is well known that the powers A^k of $A \in \mathbb{C}^{n\times n}$ tend to zero as $k \to \infty$ if $\rho(A) < 1$, where ρ is the spectral radius. However, this simple statement does not tell the whole story. Figure 17.1 plots the 2-norms of the first 30 powers of a certain 3 × 3 matrix with $\rho(A) = 0.75$. The powers do eventually decay, but initially they grow rapidly. (In this and other similar plots, k on the x-axis is plotted against $\|fl(A^k)\|_2$ on the y-axis, and the norm values are joined by straight lines for plotting purposes.) Figure 17.2 plots the 2-norms of the first 250 powers of a 14 × 14 nilpotent matrix C_{14} discussed by Trefethen and Trummer [1020, 1987] (see §17.2 for details). The plot illustrates the statement of these authors that the matrix is not power bounded in floating point arithmetic, even though its 14th power should be zero.

These examples suggest two important questions.

- For a matrix with $\rho(A) < 1$, how does the sequence $\{\|A^k\|\}$ behave? In particular, what is the size of the "hump" $\max_k \|A^k\|$?

- What conditions on A ensure that $fl(A^k) \to 0$ as $k \to \infty$?

We examine these questions in the next two sections.

17.1. Matrix Powers in Exact Arithmetic

In exact arithmetic the limiting behaviour of the powers of $A \in \mathbb{C}^{n\times n}$ is determined by A's eigenvalues. As already noted, if $\rho(A) < 1$ then $A^k \to 0$ as $k \to \infty$ if $\rho(A) > 1$, $\|A^k\| \to \infty$ as $k \to \infty$. In the remaining case of $\rho(A) = 1$, $\|A^k\| \to \infty$ if A has a defective eigenvalue λ such that $|\lambda| = 1$; A^k does not converge if A has a nondefective eigenvalue $\lambda \ne 1$ such that $|\lambda| = 1$ (although the norms of the powers may converge); otherwise, the only eigenvalue of modulus 1 is the nondefective eigenvalue 1 and A^k converges to a nonzero matrix. These statements are easily proved using the Jordan canonical form

$$A = XJX^{-1} \in \mathbb{C}^{n\times n}, \tag{17.1a}$$

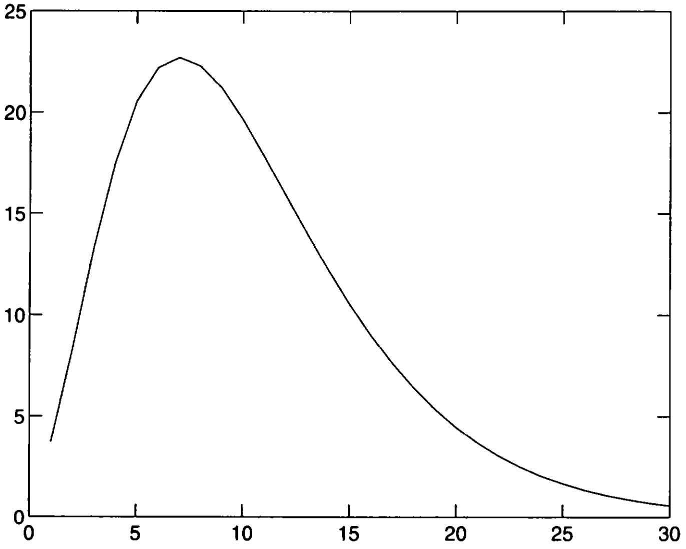


Figure 17.1. *A typical hump for a convergent, nonnormal matrix.*

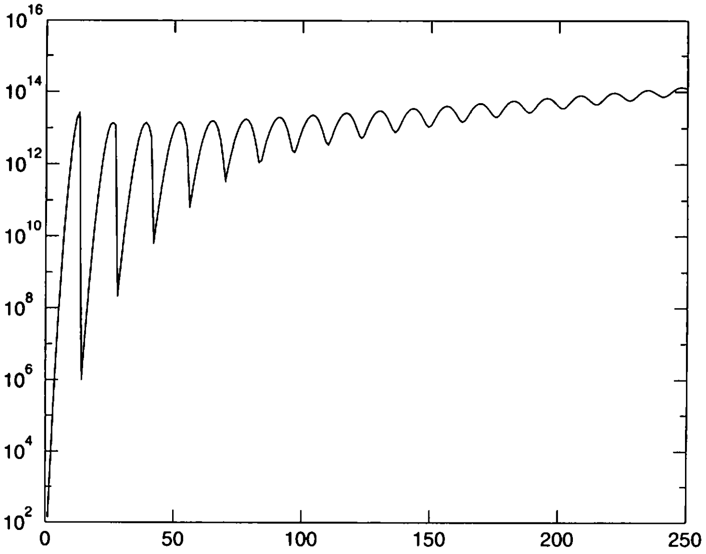


Figure 17.2. *Diverging powers of a nilpotent matrix,* C_{14}.

where X is nonsingular and

$$J = \operatorname{diag}(J_1, J_2, \ldots, J_s), \qquad J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \ddots & \ddots & \\ & & \lambda_i & 1 \\ & & & \lambda_i \end{bmatrix} \in \mathbb{C}^{n_i \times n_i}, \tag{17.1b}$$

where $n_1 + n_2 + \ldots + n_s = n$. We will call a matrix for which $A^k \to 0$ as $k \to \infty$ (or equivalently, $\rho(A) < 1$) a *convergent matrix.*

The norm of a convergent matrix can be arbitrarily large, as is shown trivially by the scaled Jordan block

$$J = \lambda \begin{bmatrix} 1 & \alpha \\ 0 & 1 \end{bmatrix}, \tag{17.2}$$

with $|\lambda| < 1$ and $\alpha \gg 1$. While the spectral radius determines the asymptotic rate of growth of matrix powers, the norm influences the initial behaviour of the powers. The interesting result that $\rho(A) = \lim_{k\to\infty} \|A^k\|^{1/k}$ for any norm (see Horn and Johnson [580, 1985, p. 299], for example) confirms the asymptotic role of the spectral radius. This formula for $\rho(A)$ has actually been considered as a means for computing it; see Wilkinson [1089, 1965, pp. 615–617] and Friedland [408, 1991].

An important quantity is the "hump" $\max_k \|A^k\|$, which can be arbitrarily large for a convergent matrix. Figure 17.1 shows the hump for the 3 × 3 upper triangular matrix with diagonal entries 3/4 and off-diagonal entries 2; this matrix has 2-norm 3.57. The shape of the plot is typical of that for a convergent matrix with norm bigger than 1. Note that if A is normal (so that in (17.1a) J is diagonal and X can be taken to be unitary) we have $\|A^k\|_2 = \|\operatorname{diag}(\lambda_i^k)\|_2 = \|A\|_2^k = \rho(A)^k$, so the problem of bounding $\|A^k\|$ is of interest only for nonnormal matrices. The hump phenomenon arises in various areas of numerical analysis. For example, it is discussed for matrix powers in the context of stiff differential equations by D. J. Higham and Trefethen [529, 1993], and by Moler and Van Loan [775, 1978] for the matrix exponential e^{At} with $t \to \infty$.

More insight into the behaviour of matrix powers can be gained by considering the 2 × 2 matrix (17.2) with $0 \le \lambda < 1$ and $\alpha > 0$. We have

$$J^k = \lambda^k \begin{bmatrix} 1 & k\alpha \\ 0 & 1 \end{bmatrix}$$

and

$$\frac{\|J^{k+1}\|_\infty}{\|J^k\|_\infty} = \lambda \frac{1 + (k+1)\alpha}{1 + k\alpha} \approx \lambda\Big(1 + \frac{1}{k}\Big) \quad \text{for large } k. \tag{17.3}$$

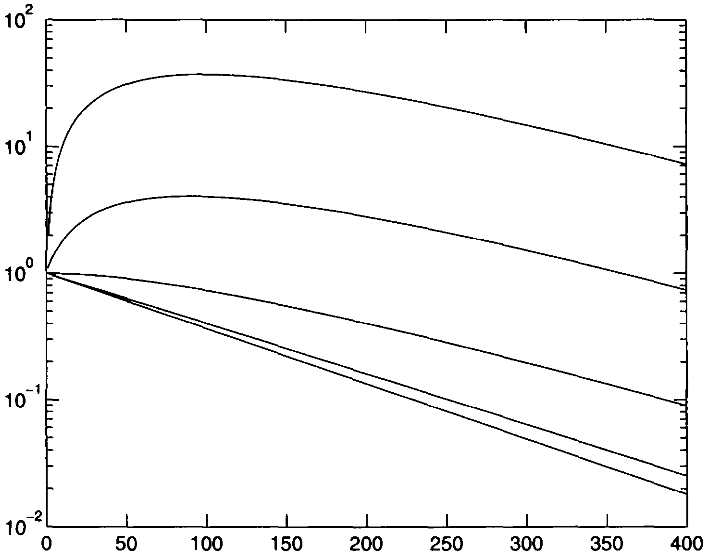


Figure 17.3. *Infinity norms of powers of* 2 × 2 *matrix J in* (17.2), *for* $l = 0.99$ *and* $a = 0$ *(bottom line) and* $a = 10^{-k}$, $k = 0{:}3$.

Hence

$$\|J^{k+1}\|_\infty < \|J^k\|_\infty \quad \text{if} \quad k > \frac{\lambda(1+\alpha)-1}{(1-\lambda)\alpha}.$$

It follows that the norms of the powers can increase for arbitrarily many steps until they ultimately decrease. Moreover, because k^{-1} tends to zero quite slowly as $k \to \infty$, the rate of convergence of $\|J^k\|_\infty$ to zero can be much slower than the convergence of λ^k to zero (see (17.3)) when λ is close to 1. In other words, nontrivial Jordan blocks retard the convergence to zero.

For this 2 × 2 matrix, the hump $\max_k \|J^k\|_\infty$ is easily shown to be approximately

$$\rho^{1/\alpha - 1/\log_e \rho} \frac{\alpha}{\log_e \rho},$$

where $\rho = \lambda^{-1} > 1$, this value being attained for $k \approx (\log_e \rho)^{-1} - \alpha^{-1}$. Figure 17.3 displays the norms of the first 400 powers of the matrices with $\lambda = 0.99$ and $a = 0, 0.001, 0.01, 0.1, 1$. The size and location of the hump are complicated expressions even in this simple case. When we generalize to direct sums of larger Jordan blocks and incorporate a similarity transformation, giving (17.1a), the qualitative behaviour of the powers becomes too difficult to describe precisely.

In the rest of this section we briefly survey bounds for $\|A^k\|$. First, however, we comment on the condition number $\kappa(X) = \|X\|\,\|X^{-1}\|$ that appears in various bounds in this chapter. The matrix X in the Jordan form (17.1a) is by no means unique [413, 1959, pp. 220–221], [467, 1976]: if A has distinct eigenvalues (hence J is diagonal) then X can be replaced by XD, for any nonsingular diagonal D, while if A has repeated eigenvalues then X can be replaced by XT, where T is a block matrix with block structure conformal with that of J and which contains some arbitrary upper trapezoidal Toeplitz blocks. We adopt the convention that $\kappa(X)$ denotes the minimum possible value of $\kappa(X)$ over all possible choices of X. This minimum value is not known for general A, and the best we can hope is to obtain a good estimate of it. However, if A has distinct eigenvalues then the results in Theorems 7.5 and 7.7 on diagonal scalings are applicable and enable us to determine (an approximation to) the minimal condition number. Explicit expressions can be given for the minimal 2-norm condition number for $n = 2$; see Young [1122, 1971, §3.8].

A trivial bound is $\|A^k\| \le \|A\|^k$. A sharper bound can be derived in terms of the *numerical radius*

$$r(A) = \max\left\{ \left|\frac{z^*Az}{z^*z}\right| : 0 \ne z \in \mathbb{C}^n \right\},$$

which is the point of largest modulus in the field of values of A. It is not hard to show that $\|A\|_2/2 \le r(A) \le \|A\|_2$ [580, 1985, p. 331]. The (nontrivial) inequality $r(A^k) \le r(A)^k$ [580, 1985, p. 333] leads to the bound

$$\|A^k\|_2 \le 2\,r(A)^k.$$

If A is diagonalizable then, from (17.1a), we have the bound

$$\|A^k\|_p \le \kappa_p(X)\rho(A)^k, \tag{17.4}$$

for any p-norm. (Since $p(A) \le \|A\|$ for any norm, we also have the lower bound $p(A)^k \le \|A^k\|_p$.) This bound is unsatisfactory for two reasons. First, by choosing A to have well-conditioned large eigenvalues and ill-conditioned small eigenvalues we can make the bound arbitrarily pessimistic (see Problem 17.1). Second, it models norms of powers of convergent matrices as monotonically decreasing sequences, which is qualitatively incorrect if there is a large hump.

The Jordan canonical form can also be used to bound the norms of the powers of a defective matrix. If XJX^{-1} is the Jordan canonical form of $\delta^{-1}A$ then

$$\|A^k\|_p \le \kappa_p(X)(\rho(A)+\delta)^k, \tag{17.5}$$

for all $\delta > 0$. This is a special case of a result of Ostrowski [812, 1973, Thin. 20.1] and the proof is straightforward: We can write $\delta^{-1}A = X(\delta^{-1}D + M)X^{-1}$, where $D = \operatorname{diag}(\lambda_i)$ and M is the off-diagonal part of the Jordan form. Then $A = X(D + \delta M)X^{-1}$, and (17.5) follows by taking norms. An alternative way of writing this bound is

$$\|A^k\|_p \le \kappa_p(X)\kappa_p(D)(\rho(A)+\delta)^k,$$

where $A = XJX^{-1}$ and $D = \mathrm{diag}(\delta^{n-1}, \delta^{n-2}, \ldots, 1)$. Note that this is not the same X as in (17.5): multiplying A by a scalar changes $\kappa(X)$ when A is not diagonalizable. Both bounds suffer from the same problems as the bound (17.4) for diagonalizable matrices.

Another bound in terms of the Jordan canonical form (17.1) of A is given by Gautschi [430, 1953]. For convergent matrices, it can be written in the form

$$\|A^k\|_F \le c\,k^{p-1}\rho(A)^k, \tag{17.6}$$

where $p = \max\{n_i : \lambda_i \ne 0\}$ and c is a constant depending only on A (c is not defined explicitly in [430, 1953]). The factor k^{p-1} makes this bound somewhat more effective at predicting the shapes of the actual curve than (17.5), but again c can be unsuitably large.

Since the norm estimation problem is trivial for normal matrices, it is natural to look for bounds that involve a measure of nonnormality. Consider the Schur decomposition $Q^*AQ = D+N$, where N is strictly upper triangular, and let S represent the set of all such N. The nonnormality of A can be measured by Henrici's *departure from normality* [516, 1962]

$$\Delta(A, \|\cdot\|) \equiv \Delta(A) = \min_{N\in S} \|N\|.$$

For the Frobenius norm, Henrici shows that $\|N\|_F$ is independent of the particular Schur form and that

$$\Delta_F(A) = \Bigl(\|A\|_F^2 - \sum_i |\lambda_i|^2\Bigr)^{1/2} \le \Bigl(\frac{n^3-n}{12}\Bigr)^{1/4} \|A^*A - AA^*\|_F^{1/2}.$$

László [690, 1994] has recently shown that $D_F(A)$ is within a constant factor of the distance from A to the nearest normal matrix:

$$\frac{\Delta_F(A)}{\sqrt{n}} \le \nu(A) \le \Delta_F(A),$$

where $v(A) = \min\{\|E\|_F : A + E \text{ is normal}\}$. Henrici uses the departure from normality to derive the 2-norm bounds

$$\|A^k\|_2 \le \begin{cases} \displaystyle\sum_{i=0}^{n-1} \binom{k}{i} \rho(A)^{k-i}\Delta_2(A)^i, & \rho(A) > 0, \\ \Delta_2(A)^k, & \rho(A) = 0 \text{ and } k < n. \end{cases} \tag{17.7}$$

Empirical evidence suggests that the first bound in (17.7) can be very pessimistic. However, for normal matrices both the bounds are equalities.

Another bound involving nonnormality is given by Golub and Van Loan [470, 1989, Lem. 7.3.2]. They show that, in the above notation,

$$\|A^k\|_2 \le (1+\theta)^{n-1}\left(\rho(A) + \frac{\Delta_F(A)}{1+\theta}\right)^k$$

for any $q \ge 0$. This bound is an analogue of (17.5) with the Schur form replacing the Jordan form. Again, there is equality when A is normal (if we set $q = 0$).

To compare bounds based on the Schur form with ones based on the Jordan form we need to compare $D(A)$ with $\kappa(X)$. If A is diagonalizable then [710, 1969, Thin. 4]

$$\kappa_2(X) \ge \left(1 + \frac{\Delta_F(A)^2}{\|A\|_F^2}\right)^{1/2};$$

it can be shown by a 2 × 2 example that $\min_X \kappa_2(X)$ can exceed $D_F(A)/\|A\|_F$ by an arbitrary factor [201, 1993, §4.2.7], [190, 1996, §9.1.1].

Another tool that can be used to bound the norms of powers is the pseudospectrum of a matrix, popularized by Trefethen [1017, 1992], [1018]. The *ϵ-pseudospectrum* of $A \in \mathbb{C}^{n\times n}$ is defined, for a given $\epsilon > 0$, to be the set

$$\Lambda_\epsilon(A) = \{\, z : z \text{ is an eigenvalue of } A + E \text{ for some } E \text{ with } \|E\|_2 \le \epsilon \,\},$$

and it can also be represented, in terms of the resolvent $(zI - A)^{-1}$, as

$$\Lambda_\epsilon(A) = \{\, z : \|(zI - A)^{-1}\|_2 \ge \epsilon^{-1} \,\}.$$

As Trefethen notes [1017, 1992], by using the Cauchy integral representation of A^k (which involves a contour integral of the resolvent) one can show that

$$\|A^k\|_2 \le \epsilon^{-1}\rho_\epsilon(A)^{k+1}, \tag{17.8}$$

where the *ϵ-pseudospectral radius*

$$\rho_\epsilon(A) = \max\{\, |z| : z \in \Lambda_\epsilon(A) \,\}. \tag{17.9}$$

This bound is very similar in flavour to (17.5). The difficulty is transferred from estimating $\kappa(X)$ to choosing ϵ and estimating $\rho_\epsilon(A)$.

Bai, Demmel, and Gu [39, 1994] consider A with $p(A) < 1$ and manipulate the Cauchy integral representation of A^k in a slightly different way from Trefethen to produce a bound in terms of the distance to the nearest unstable matrix,

$$\begin{aligned} d(A) &= \min\{\, \|\Delta A\|_2 : A + \Delta A \text{ has an eigenvalue of modulus } 1 \,\} \\ &= \min\{\, \epsilon : \rho_\epsilon(A) = 1 \,\}. \end{aligned}$$

Their bound is

$$\|A^k\|_2 \le \begin{cases} m\alpha_m(1-d(A))^m, & m > (1-d(A))/d(A), \\ 1/d(A), & m \le (1-d(A))/d(A), \end{cases}$$

where $e \le a_m := (1+1/m)^{m+1} \le 4$. Note that $d(A) \le 1$ when $p(A) < 1$, *as* is easily seen from the Schur decomposition. The distance $d(A)$ is not easy to compute. One approach is a bisection technique of Byers [175, 1988].

Finally, we mention that the Kreiss matrix theorem provides a good estimate of $\sup_{k>0} \|A^k\|$ for a general $A \in \mathbb{C}^{n\times n}$, albeit in terms of an expression that involves–the resolvent and is not easy to compute:

$$\phi(A) \le \sup_{k\ge 0} \|A^k\|_2 \le n\, e\, \phi(A),$$

where $\phi(A) = \sup\{(|z|-1)\|(zI-A)^{-1}\|_2 : |z| > 1\}$ and $e = \exp(1)$. Details and references are given by Wegert and Trefethen [1071, 1994].

17.2. Bounds for Finite Precision Arithmetic

The formulae $A \cdot A^k$ or $A^k \cdot A$ can be implemented in several ways, corresponding to different loop orderings in each individual product, but as long as each product is formed using the standard formula $(AB)_{ij} = \sum_k a_{ik}b_{kj}$, all these variations satisfy the same rounding error bounds. We do not analyse here the use of the binary powering technique, where, for example, A^9 is formed as $A((A^2)^2)^2$; alternate multiplication on the left and right $(fl(A^k) = fl(Afl(A^{k-2})A))$; or fast matrix multiplication techniques such as Strassen's method. None of these methods is equivalent to repeated multiplication in finite precision arithmetic.

We suppose, without loss of generality, that the columns of A^m are computed one at a time, the jth as $fl(A(A(\dots(Ae_j)\dots)))$, where e_j is the jth unit vector. The error analysis for matrix–vector multiplication shows that the jth computed column of Am satisfies

$$fl(A^m e_j) = (A+\Delta A_1)(A+\Delta A_2)\dots(A+\Delta A_m)e_j, \tag{17.10}$$

where, for both real and complex matrices, we have (Problem 3.7)

$$|\Delta A_i| \le \gamma_{n+2}|A|. \tag{17.11}$$

It follows that

$$|fl(A^m e_j)| \le (1+\gamma_{n+2})^m |A|^m e_j,$$

and so a sufficient condition for convergence of the computed powers is that

$$\rho(|A|) < \frac{1}{1+\gamma_{n+2}}. \tag{17.12}$$

This result is useful in certain special cases: $\rho(|A|) = \rho(A)$ if A is triangular or has a checkerboard sign pattern (since then $|A| = DAD^{-1}$ where $D = \mathrm{diag}(\pm 1)$); if A is normal then $\rho(|A| \le \sqrt{n}\rho(A)$ (this bound being attained for a Hadamard matrix); and in Markov processes, where the a_{ij} are transition probabilities, $|A| = A$. However, in general $\rho(|A|)$ can exceed $\rho(A)$ by an arbitrary factor (see Problem 17.2).

To obtain sharper and more informative results it is necessary to use more information about the matrix. In the following theorem we give a sufficient condition, based on the Jordan canonical form, for the computed powers of a matrix to converge to zero. Although the Jordan form is usually eschewed by numerical analysts because of its sensitivity to perturbations, it is convenient to work with in this application and leads to an informative result.

Theorem 17.1 (Higham and Knight). *Let $A \in \mathbb{C}^{n\times n}$ with the Jordan form (17.1) have spectral radius $\rho(A) < 1$. A sufficient condition for $fl(A^m) \to 0$ as $m \to \infty$ is*

$$4t\gamma_{n+2}\kappa_\infty(X)\|A\|_\infty < (1-\rho(A))^t, \tag{17.13}$$

where $t = \max_i n_i$.

Proof. It is easy to see that if we can find a nonsingular matrix S such that

$$\|S^{-1}AS\|_\infty + \kappa_\infty(S)\|\Delta A_i\|_\infty < 1 \tag{17.14}$$

for all i, then the product

$$(A+\Delta A_1)\dots(A+\Delta A_m) = S(S^{-1}AS+S^{-1}\Delta A_1 S)\dots(S^{-1}AS+S^{-1}\Delta A_m S)S^{-1}$$

tends to 0 as $m \to \infty$. In the rest of the proof we construct such a matrix S for the ΔA_i in (17.10).

Let $P(\epsilon) = \mathrm{diag}(P_1(\epsilon),\dots,P_s(\epsilon))$ where $0 < \epsilon < 1-\rho(A)$ and

$$P_i(\epsilon) = \mathrm{diag}((1-|\lambda_i|-\epsilon)^{1-n_i}, (1-|\lambda_i|-\epsilon)^{2-n_i},\dots,1) \in \mathbb{R}^{n_i\times n_i}.$$

Now consider the matrix $P(\epsilon)^{-1}JP(\epsilon)$. Its ith diagonal block is of the form $\lambda_i I + (1-|\lambda_i|-\epsilon)N$, where the only nonzeros in N are 1s on the first superdiagonal, and so

$$\|P(\epsilon)^{-1}X^{-1}AXP(\epsilon)\|_\infty = \|P(\epsilon)^{-1}JP(\epsilon)\|_\infty \le \max(|\lambda_i|+1-|\lambda_i|-\epsilon) = 1-\epsilon.$$

Defining $S = XP(\epsilon)$, we have $\|S^{-1}AS\|_\infty \le 1-\epsilon$ and

$$\kappa_\infty(S) \le \kappa_\infty(P(\epsilon))\kappa_\infty(X) \le (1-\rho(A)-\epsilon)^{1-t}\kappa_\infty(X). \tag{17.15}$$

Now we set $\epsilon = \theta(1-\rho(A))$ where $0 < \theta < 1$ and we determine θ so that (17.14) is satisfied, that is, so that $\kappa_\infty(S)\|\Delta A_i\|_\infty < \epsilon$ for all i. From (17.11) and (17.15) we have

$$\kappa_\infty(S)\|\Delta A_i\|_\infty \le \gamma_{n+2}(1-\theta)^{1-t}(1-\rho(A))^{1-t}\kappa_\infty(X)\|A\|_\infty.$$

Therefore (17.14) is satisfied if

$$\gamma_{n+2}(1-\theta)^{1-t}(1-\rho(A))^{1-t}\kappa_\infty(X)\|A\|_\infty < \theta(1-\rho(A)),$$

that is, if

$$\gamma_{n+2}\kappa_\infty(X)\|A\|_\infty < (1-\theta)^{t-1}\theta(1-\rho(A))^t.$$

If the integer t is greater than 1 then the function $f(q) = (1-q)^{t-1}q$ has a maximum on [0,1] at $q_* = t^{-1}$ and $f(q_*) = (t-1)^{-1}(1-t^{-1})^t$ satisfies $(4(t-1))^{-1} \le f(q_*) < e^{-1}$. We conclude that for all integers $1 < t < n$.

$$\gamma_{n+2}\kappa_\infty(X)\|A\|_\infty < \frac{1}{4t}(1-\rho(A))^t$$

is sufficient to ensure that (17.14) holds.

If A is normal then $||A||_2 = p(A) < 1$, $t = 1$, and $\kappa_2(X) = 1$, so (17.13) can be written in the form

$$\rho(A) < \frac{1}{1+c_n u},$$

where c_n is a constant depending on n. This condition is also easily derived by taking 2-norms in (17.10) and (17.11).

We can show the sharpness of the condition in Theorem 17.1 by using the Chebyshev spectral differentiation matrix $C_n \in \mathbb{R}^{n\times n}$ described by Trefethen and Trummer [1020, 1987]. The matrix C_n arises from degree $n-1$ polynomial interpolation of n arbitrary data values at n Chebyshev points, including a boundary condition at 1. It is nilpotent and is similar to a single Jordan block of dimension n. We generate C_n in MATLAB using the routine chebspec from the Test Matrix Toolbox (see Appendix E). Figure 17.4 shows the norms of the powers of four variants of the C_n matrix.

The powers of C_8 converge to zero, while the powers of $15C_8$ diverge. Using a technique for estimating $k_2(X)$ described in [565, 1995], we find that $u\kappa_2(X)\|C_8\|_2 \approx 1.08 \times 10^{-9}$, which is safely less than 1, so that Theorem 17.1 predicts convergence. For $15C_8$ we have $u\kappa_2(X)\|15C_8\|_2$ 2.7, so the theorem correctly does not predict convergence.

Next, for the matrix $A = C_{13} + 0.36I$, whose powers diverge, we have $u\kappa_2(X)\|A\|_2/(1-\rho(A))^{13} \approx 13.05$, and for $A = C_{13} + 0.01I$, whose powers converge, $u\kappa_2(X)\|A\|_2/(1-\rho(A))^{13} \approx 0.01$, so again the theorem is reasonably sharp.

The plots reveal interesting scalloping patterns in the curves of the norms. For C_8 and $15C_8$ the dips are every 8 powers, but the point of first dip and the dipping intervals are altered by adding different multiples of the identity matrix, as shown by the C_{13} examples. Explaining this behaviour is an open problem (see Problem 17.3).

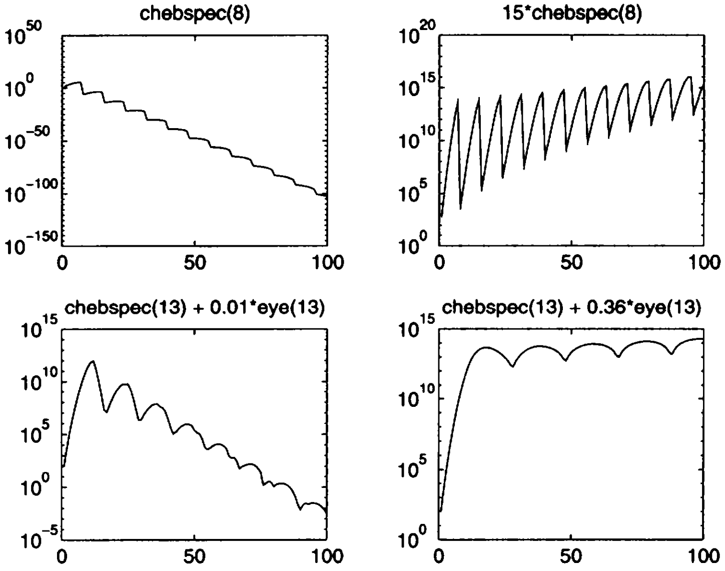


Figure 17.4. *Computed powers of* chebspec *matrices.*

We saw in the last section that the powers of A can be bounded in terms of the pseudospectral radius. Can the pseudospectrum provide information about the behaviour of the computed powers? Figure 17.5 shows approximations to the **ϵ-pseudospectra** for the matrices used in Figure 17.4, where $\epsilon = n\|A\|_2 u$; the (computed) eigenvalues are plotted as crosses "×". We see that the pseudospectrum lies inside the unit disc precisely when the powers converge to zero.

A heuristic argument based on (17.10) and (17.11) suggests that, if for randomly chosen perturbations DA_i with $\|DA_i\| \le c_n u\|A\|$, most of the eigenvalues of the perturbed matrices lie outside the unit disc, then we can expect a high percentage of the terms $A + DA_i$ in (17.10) to have spectral radius bigger than 1 and hence we can expect the product to diverge. On the other hand, if the $c_n u\|A\|$-pseudospectrum is wholly contained within the unit disc, each $A + DA_i$ will have spectral radius less than 1 and the product can be expected to converge. (Note, however, that if $p(A) < 1$ and $p(B) < 1$ it is not necessarily the case that $p(AB) < 1$.) To make this heuristic precise, we need an analogue of Theorem 17.1 phrased in terms of the pseudospectrum rather than the Jordan form.

Theorem 17.2 (Higham and Knight). *Suppose that* $A \in \mathbb{C}^{n\times n}$ *is diagonalizable with* $A = X\,\mathrm{diag}(\lambda_i)X^{-1}$ *and has a unique eigenvalue* λ_1 *of largest modulus. Suppose that* $\|X\|_1 = \sum_{i=1}^{n} |x_{i1}|$ *and* $\|X^{-1}\|_\infty = \sum_{j=1}^{n} |y_{1j}|$, *where*

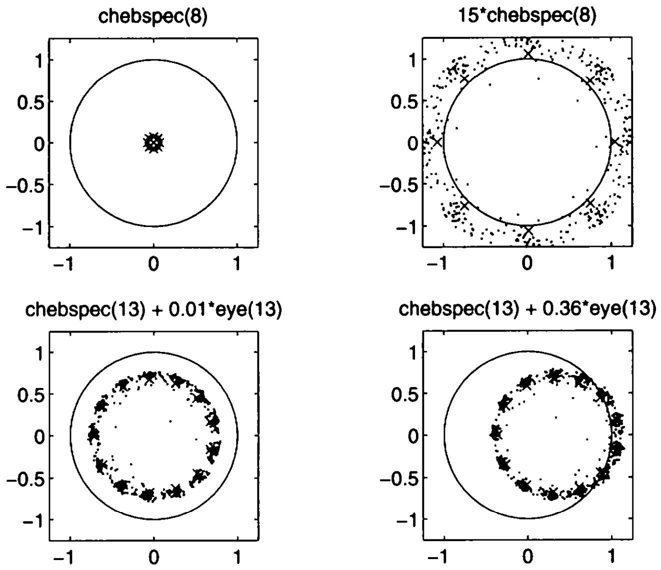


Figure 17.5. *Pseudospectra for* chebspec *matrices.*

$X^{-1} = (y_{ij})$. *If* $\rho_\epsilon(A) < 1$ *for* $\epsilon = c_n u\|A\|_2$, *where* c_n *is a constant depending only on n, then, provided that a certain* $O(\epsilon^2)$ *term can be ignored,* $\lim_{m\to\infty} fl(A^m) = 0$.

Proof. It can be shown (see [565, 1995]) that the conditions on $\|X\|_1$ and $\|X^{-1}\|_\infty$ imply there is a perturbation $\tilde{A} = A + DA$ of A with $\|DA\|_2 = \epsilon$ such that

$$\rho(\tilde{A}) \geq \rho(A) + \frac{\kappa_2(X)\epsilon}{n^2} + O(\epsilon^2).$$

Hence, if $\rho_\epsilon(A) < 1$ then $\rho(A) + \kappa_2(X)\epsilon/n^2 < 1 + O(\epsilon^2)$. Ignoring the $O(\epsilon^2)$ term and rearranging gives

$$c_n u \kappa_2(X)\|A\|_2/n^2 < 1 - \rho(A).$$

Using Theorem 17.1 we have the required result for $c_n = 4n^2(n + 2)$, since $t = 1$. □

Suppose we compute the eigenvalues of A by a backward stable algorithm, that is, one that yields the exact eigenvalues of $A+E$, where $\|E\|_2 \leq c_n u\|A\|_2$, with c_n a modest constant. (An example of such an algorithm is the *QR* algorithm [470, 1989, §7.5]). Then the computed spectral radius $\widehat{\rho}$ satisfies $\widehat{\rho} < \rho_{c_n u\|A\|_2}(A)$. In view of Theorem 17.2 we can formulate a rule of thumb-one that bears a pleasing symmetry with the theoretical condition for convergence:

> The computed powers of A can be expected to converge to 0 if the spectral radius computed via a backward stable eigensolver is less than 1.

This rule of thumb has also been discussed by Trefethen and Trummer [1020, 1987] and Reichel and Trefethen [866, 1992]. In our experience the rule of thumb is fairly reliable when $\widehat{\rho}$ is not too close to 1. For the matrices used in our examples we have, using MATLAB'S eig function,

$$\widehat{\rho}(C_8) = 0.073, \qquad \widehat{\rho}(15C_8) = 2.7, \qquad \widehat{\rho}(C_{14}) = 1.005,$$
$$\widehat{\rho}(C_{13} + 0.01I) = 0.70, \quad \widehat{\rho}(C_{13} + 0.36I) = 1.05,$$

and we observed convergence of the computed powers for C_8 and $C_{13} + 0.01I$ and divergence for the other matrices.

17.3. Application to Stationary Iteration

As we saw in the previous chapter, the errors in stationary iteration satisfy $e_k = (M^{-1}N)^k e_0$, so convergence of the iteration depends on the convergence of $(M^{-1}N)^k$ to zero as $k \to \infty$. While the errors in stationary iteration are not precisely modelled by the errors in matrix powering, because matrix powers are not formed explicitly, the behaviour of the computed powers $fl((M^{-1}N)^k)$ can be expected to give some insight into the behaviour of stationary iteration.

For the successive overrelaxation (SOR) example at the start of Chapter 16, the matrix $G = M^{-1}N$ is lower triangular with $g_{ij} = 0.5(-1)^{i-j}$. The computed powers of G in MATLAB reach a maximum norm of $\approx 10^{28}$ at $k = 99$ and then decay to zero; the eventual convergence is inevitable in view of the condition (17.12), which clearly is satisfied for this triangular G. An approximation to the $u\|G\|_2$-pseudospectrum is plotted in Figure 17.6, and we see clearly that part of the pseudospectrum lies outside the unit disk. These facts are consistent with the nonconvergence of the SOR iteration (see Figure 16.1).

That the pseudospectrum of G gives insight into the behaviour of stationary iteration has also been observed by Trefethen [1015, 1990], [1017, 1992], [1018] and Chatelin and Frayssé [203, 1992], but no rigorous results about the connection are available.

17.4. Notes and References

This chapter is based closely on Higham and Knight [565, 1995].

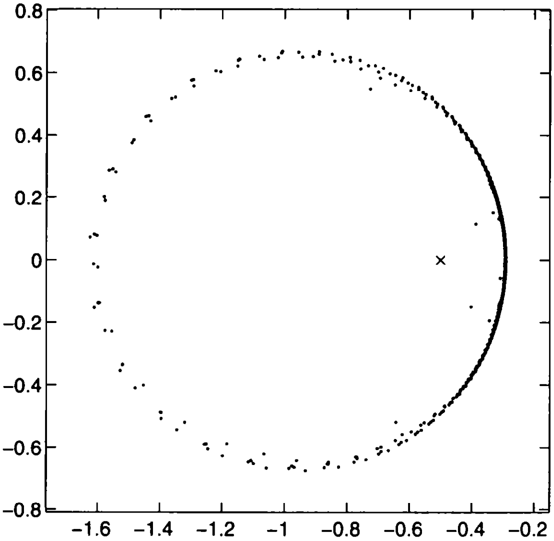


Figure 17.6. *Pseudospectrum for SOR iteration matrix.*

The analysis for the powers of the matrix (17.2) is modelled on that of Stewart [953, 1994], who uses the matrix to construct a Markov chain whose second largest eigenvalue does not correctly predict the decay of the transient.

For some results on the asymptotic behaviour of the powers of a nonnegative matrix, see Friedland and Schneider [409, 1980].

Another application of matrix powering is in the scaling and squaring method for computing the matrix exponential, which uses the identity $e^A = (e^{A/m})^m$ together with a Taylor or Padé approximation to $e^{A/m}$; see Moler and Van Loan [775, 1978].

Problems

17.1. Let $A \in \mathbb{C}^{n\times n}$ be diagonalizable: $A = XLX^{-1}$, $L = \mathrm{diag}(\lambda_i)$ Construct a parametrized example to show that the bound $\|A^k\|_2 \le \kappa_2(X)\rho(A)^k$ can be arbitrarily weak.

17.2. Show that $p(|A|)/p(A)$ can be arbitrarily large for $A \in \mathbb{R}^{n\times n}$.

17.3. (RESEARCH PROBLEM) Explain the scalloping patterns in the curves of norms of powers of a matrix, as seen, for example, in Figure 17.4. (Consider exact arithmetic, as the phenomenon is not rounding error dependent.)

17.4. (RESEARCH PROBLEM) obtain a sharp sufficient condition for $fl(A^k) \to 0$ in terms of the Schur decomposition of $A \in \mathbb{C}^{n\times n}$ with $p(A) < 1$.

Previous Home Next

Chapter 18

QR Factorization

Any orthogonal matrix can be written as the product of reflector matrices.
Thus the class of reflections is rich enough for all occasions
and yet each member is characterized by a single vector
which serves to describe its mirror.

— BERESFORD N. PARLETT, *The Symmetric Eigenvalue Problem (1980)*

A key observation for understanding the numerical properties of the
modified Gram–Schmidt algorithm is that it can be interpreted as
Householder QR factorization applied to the matrix A
augmented with a square matrix of zero elements on top.
These two algorithms are not only mathematically . . .
but also numerically equivalent.
This key observation, apparently by Charles Sheffield,
was relayed to the author in 1968 by Gene Golub.

— AKE BJÖRCK, *Numerics of Gram-Schmidt Orthogonalization (1994)*

The great stability of unitary transformations in numerical analysis
springs from the fact that both the ℓ_2-norm
and the Frobenius norm are unitarily invariant.
This means in practice that even when rounding errors are made,
no substantial growth takes place in the
norms of the successive transformed matrices.

— J. H. WILKINSON,
Error Analysis of Transformations Based on the
Use of Matrices of the Form $I - 2ww^H$ (1965)

The QR factorization is a versatile computational tool that finds use in linear equation, least squares and eigenvalue problems. It can be computed in several ways, including by the use of Householder transformations and Givens rotations, and by the Gram–Schmidt method. We explore the numerical prop erties of all three methods in this chapter. We also examine the use of iterative refinement on a linear system solved with a QR factorization and consider the inherent sensitivity of the QR factorization.

18.1. Householder Transformations

A Householder matrix (also known as a Householder transformation, or Householder reflector) is a matrix of the form

$$P = I - \frac{2}{v^T v} v v^T, \qquad 0 \neq v \in \mathbb{R}^n.$$

It enjoys the properties of symmetry and orthogonality, and, consequently, is involuntary ($P^2 = I$). The application of P to a vector yields

$$Px = x - \left(\frac{2v^T x}{v^T v} \right) v.$$

Figure 18.1 illustrates this formula and makes it clear why P is sometimes called a Householder reflector: it reflects x about the hyperplane $\mathrm{span}(v)^\perp$.

Householder matrices are powerful tools for introducing zeros into vectors. Consider the question "given x and y can we find a Householder matrix P such that $Px = y$?" Since P is orthogonal we clearly require that $\|x\|_2 = \|y\|_2$. Now

$$Px = y \quad \Longleftrightarrow \quad x - 2\left(\frac{v^T x}{v^T v} \right) v = y,$$

and this last equation has the form $\alpha v = x - y$ for some α. But P is independent of the scaling of v, so we can set $\alpha = 1$.

With $u = x - y$ we have

$$v^T v = x^T x + y^T y - 2x^T y,$$

and, since $x^T x = y^T y$,

$$v^T x = x^T x - y^T x = \frac{1}{2} v^T v.$$

Therefore

$$Px = x - v = y,$$

as required. We conclude that, provided $\|x\|_2 = \|y\|_2$, we can find a Householder matrix P such that $Px = y$. (Strictly speaking, we have to exclude the case $x = y$, which would require $u = 0$, making P undefined).

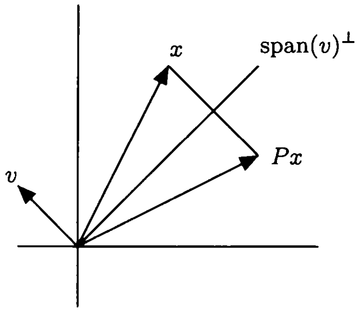


Figure 18.1. *Householder matrix P times vector x.*

Normally we choose y to have a special pattern of zeros. The usual choice is $y = se_1$ where $s = \pm\|x\|_2$, which yields the maximum number of zeros in y. Then

$$v = x - y = x - \sigma e_1.$$

We choose $\mathrm{sign}(s) = -\mathrm{sign}(x_1)$ to avoid cancellation in the expression for u_1.

18.2. *QR* Factorization

A *QR* factorization of $A \in \mathbb{R}^{m\times n}$ with $m \geq n$ is a factorization

$$A = QR = [Q_1 \quad Q_2] \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1,$$

where $Q \in \mathbb{R}^{m\times m}$ is orthogonal and $R_1 \in \mathbb{R}^{n\times n}$ is upper triangular. The matrix R is called upper trapezoidal, since the term triangular applies only to square matrices. Depending on the context, either the full factorization $A = QR$ or the "economy size" version $A = Q_1 R_1$ can be called a *QR* factorization. A quick existence proof of the *QR* factorization is provided by the Cholesky factorization: if A has full rank and $A^T A = R^T R$ is a Cholesky factorization, then $A = AR^{-1} \cdot R$ is a *QR* factorization. The *QR* factorization is unique if A has full rank and we require R to have positive diagonal elements ($A = QD \cdot DR$ is a *QR* factorization for any $D = \mathrm{diag}(\pm 1)$).

The *QR* factorization can be computed by premultiplying the given matrix by a suitably chosen sequence of Householder matrices. The process is illustrated for a generic 4 × 3 matrix as follows:

$$A = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{P_1} \left[\begin{array}{c|cc} \times & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{array}\right] \xrightarrow{P_2}$$

$$\left[\begin{array}{cc|c} \times & \times & \times \\ 0 & \times & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \times \end{array}\right] \xrightarrow{P_3} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix} = R.$$

The general process is adequately described by the kth stage of the reduction to triangular form. With $A_1 = A$ we have, at the start of the kth stage,

$$A_k = \begin{bmatrix} R_{k-1} & z_k & B_k \\ 0 & x_k & C_k \end{bmatrix}, \quad R_{k-1} \in \mathbb{R}^{(k-1)\times(k-1)}, \quad x_k \in \mathbb{R}^{m-k+1}, \quad (18.1)$$

where R_{k-1} is upper triangular. Choose a Householder matrix $\widetilde{P}_k$ such that $\widetilde{P}_k x_k = \sigma e_1$ and embed $\widetilde{P}_k$ into an $m \times m$ matrix

$$P_k = \begin{bmatrix} I_{k-1} & 0 \\ 0 & \widetilde{P}_k \end{bmatrix}. \qquad (18.2)$$

Then let $A_{k+1} = P_k A_k$. Overall, we obtain $R = P_n P_{n-1} \dots P_1 A =: Q^T A$ ($P_n = I$ *if* $m = n$).

To compute A_{k+1} we need to form $\widetilde{P}C_k$. We can write

$$\widetilde{P}_k C_k = (I - \beta v v^T) C_k = C_k - \beta v (v^T C_k), \qquad \beta = 2/(v^T v),$$

which shows that the matrix product can be formed as a matrix–vector product followed by an outer product. This approach is much more efficient than forming $\widetilde{P}_k$ explicitly and doing a matrix multiplication.

The overall cost of the Householder reduction to triangular form is $2n^2(m - n/3)$ flops. The explicit formation of Q requires a further $4(m^2 n - mn^2 + n^3/3)$ flops, but for many applications it suffices to leave Q in factored form.

18.3. Error Analysis of Householder Computations

It is well known that computations with Householder matrices are very stable. Wilkinson showed that the computation of a Householder vector, and the application of a Householder matrix to a given matrix, are both normwise stable, in the sense that the computed Householder vector is very close to the exact one and the computed update is the exact update of a tiny normwise perturbation of the original matrix [1089, 1965, pp. 153–162, 236], [1090, 1965]. Wilkinson also showed that the Householder *QR* factorization algorithm is normwise backward stable [1089, p. 236]. In this section we give a combined componentwise and normwise error analysis of Householder matrix computations. The componentwise bounds provide extra information over the normwise ones that is essential in certain applications (for example, the analysis of iterative refinement).

Lemma 18.1. *Let $x \in \mathbb{R}^n$. Consider the following construction of $\beta \in \mathbb{R}$ and $v \in \mathbb{R}^n$ such that $Px = se_1$, where $P = I - bvv^T$ is a Householder matrix with $b = 2/(v^Tv)$:*

```
$v = x$
$s = \mathrm{sign}(x_1)\|x\|_2$    % $\sigma = -s$
$v_1 = v_1 + s$
$\beta = 1/(sv_1)$
```

In floating point arithmetic the computed $\widehat{\beta}$ and $\widehat{v}$ satisfy $\widehat{v}(2\colon n) = v(2\colon n)$ and

$$\widehat{\beta} = \beta(1+\theta_{4n+8}),$$
$$\widehat{v}_1 = v_1(1+\theta_{n+2}),$$

where $|q_k| \le g_k$.

Proof. We sketch the proof. Each occurrence of δ denotes a different number bounded by $|\delta| \le u$. We compute $fl(x^Tx) = (1+q_n)x^Tx$, and then $fl(\|x\|_2) = (1+\delta)(1+\theta_n)^{1/2}(x^Tx)^{1/2} = (1+\theta_{n+1})\|x\|_2$ (the latter term $1+q_{n+1}$ is suboptimal, but our main aim is to keep the analysis simple). Hence $\widehat{s} = (1+\theta_{n+1})s$.

For notational convenience, define $w = u_1 + s$. We have $\widehat{w} = (v_1+\widehat{s})(1+\delta) = w(1+\theta_{n+2})$ (essentially because there is no cancellation in the sum). Hence

$$\widehat{\beta} = fl(1/(\widehat{s}\widehat{w})) = \frac{(1+\delta)^2}{(1+\theta_{n+1})s(1+\theta_{n+2})w}$$
$$= \frac{(1+\delta)^2}{(1+\theta_{2n+3})sw} = (1+\theta_{4n+8})\beta. \qquad \square$$

For convenience we will henceforth write Householder matrices in the form $I - vv^T$, which requires $\|u\|_2 = \sqrt{2}$ and amounts to redefining $u := \sqrt{\beta}v$

and b := 1 in the representation of Lemma 18.1. We can then write, using Lemma 18.1,

$$\widehat{v} = v + \Delta v, \qquad |\Delta v| \le \gamma_{cm}|v| \quad (v \in \mathbb{R}^m, \ \ \|v\|_2 = \sqrt{2}), \tag{18.3}$$

where, as required for the next two results, the dimension is now m. Here, we have introduced the generic constant

$$\gamma_{cm} = \frac{cmu}{1 - cmu},$$

in which c denotes a small integer constant whose exact value is unimportant. We will make frequent use of g_{cm} in the rest of this chapter, because it is not worthwhile to evaluate the integer constants in our bounds explicitly. Because we are not "chasing constants" we can afford to be somewhat cavalier and freely use inequalities such as

$$\gamma_{cm} + \gamma_m + \gamma_{cm}\gamma_m + \gamma_{cm}^2 \le \gamma'_{cm}$$

(recall Lemma 3.3), where we signify by the prime that the constant c on the right-hand side is different from that on the left.

The next result describes the application of a Householder matrix to a vector, and is the basis of all the subsequent analysis. In the applications of interest P is defined as in Lemma 18.1, but we will allow P to be an arbitrary Householder matrix. Thus v is an arbitrary, normalized vector, and the only assumption we make is that the computed $\widehat{v}$ satisfies (18.3).

Lemma 18.2. *Let* $b \in \mathbb{R}^m$ *and consider the computation of* $y = \widehat{P}b = (I - \widehat{v}\widehat{v}^T)b = b - \widehat{v}(\widehat{v}^Tb)$, *when* $\widehat{v} \in \mathbb{R}^m$ *satisfies* (18.3): *The computed* $\widehat{y}$ *satisfies*

$$\widehat{y} = (P + \Delta P)b, \qquad \|\Delta P\|_F \le \gamma_{cm},$$

where $P = I - vv^T$.

Proof. (Cf. the proof of Lemma 3.8,) We have

$$\widehat{w} := fl(\widehat{v}(\widehat{v}^Tb)) = (\widehat{v} + \Delta\widehat{v})(\widehat{v}^T(b + \Delta b)),$$

where $|\Delta\widehat{v}| \le u|\widehat{v}|$ and $|Db| \le g_m|b|$. Hence

$$\widehat{w} = (v + \Delta v + \Delta\widehat{v})(v + \Delta v)^T(b + \Delta b) =: v(v^Tb) + \Delta w,$$

where $|Dw| \le g_{cm}|u||u^T||b|$. Then

$$\widehat{y} = fl(b - \widehat{w}) = b - v(v^Tb) - \Delta w + \Delta y_1, \qquad |\Delta y_1| \le u|b - \widehat{w}|.$$

We have

$$|-\Delta w + \Delta y_1| \le u|b| + \gamma'_{cm}|v||v^T||b|.$$

Hence $\widehat{y} = Pb + \Delta y$, where $\|\Delta y\|_2 \le \gamma''_{cm}\|b\|_2$. But then $\widehat{y} = (P + \Delta P)b$, where $\Delta P = \Delta y b^T/b^T b$ satisfies $\|\Delta P\|_F = \|\Delta y\|_2/\|b\|_2 \le \gamma''_{cm}$. $\square$

Next, we consider a sequence of Householder transformations applied to a matrix. Again, each Householder matrix is arbitrary and need have no connection to the matrix to which it is being applied. In the cases of interest, the Householder matrices P_k have the form (18.2), and so are of ever-decreasing effective dimension, but to exploit this property would not lead to any significant improvement in the bounds.

For the remaining results in this section, we make the (reasonable) assumption that an inequality of the form

$$mr\gamma_{cm} < \frac{1}{2} \tag{18.4}$$

holds, where r is the number of Householder transformations and it is implicit that G and H denote nonnegative matrices.

Lemma 18.3. *Consider the sequence of transformations*

$$A_{k+1} = P_k A_k, \qquad k = 1{:}\,r,$$

where $A_1 = A \in \mathbb{R}^{m\times n}$ and $P_k = I - v_k v_k^T \in \mathbb{R}^{m\times m}$ is a Householder matrix. Assume that the transformations are performed using computed Householder vectors $\widehat{v}_k \approx v_k$ that satisfy (18.3). The computed matrix A_{r+1} satisfies

$$\widehat{A}_{r+1} = Q^T(A + \Delta A), \tag{18.5}$$

where $Q^T = P_r P_{r-1} \dots P_1$ and DA satisfies the normwise and componentwise bounds

$$\begin{aligned} \|\Delta A\|_F &\le r\gamma_{cm}\|A\|_F, \\ |\Delta A| &\le mr\gamma_{cm} G|A|, \quad \|G\|_F = 1. \end{aligned}$$

(In fact, we can take $G = m^{-1}ee^T$, where $e = [1, 1, \dots, 1]^T$.) In the special case $n = 1$, so that $A \equiv a$, we have $\widehat{a}^{(r+1)} = (Q + \Delta Q)^T a$ with $\|\Delta Q\|_F \le r\gamma_{cm}$.

Proof. First, we consider the jth column of A, a_j, which undergoes the transformations $a_j^{(r+1)} = P_r \dots P_1 a_j$. By Lemma 18.2 we have

$$\widehat{a}_j^{(r+1)} = (P_r + \Delta P_r)\dots(P_1 + \Delta P_1)a_j,$$

where each DP_k depends on j and satisfies $\|DPk\|_F \le g_{cm}$. Using Lemma 3.6 we obtain

$$\begin{aligned} \widehat{a}_j^{(r+1)} &= Q^T(a_j + d_j), \\ \|d_j\|_2 &\le \bigl((1+\gamma_{cm})^r - 1\bigr)\|a_j\|_2 \le \frac{r\gamma_{cm}}{1 - r\gamma_{cm}}\|a_j\|_2 = r\gamma'_{cm}\|a_j\|_2, \end{aligned} \tag{18.6}$$

using Lemma 3.1 and assumption (18.4). Hence *DA* in (18.5) satisfies

$$\|\Delta A\|_F^2 = \sum_{j=1}^{n} \|d_j\|_2^2 \le r^2\gamma_{cm}'^2 \sum_{j=1}^{n} \|a_j\|_2^2 = r^2\gamma_{cm}'^2\|A\|_F^2.$$

NOW, since $||a_j||_2 \le ||a_j||_1 = e^T|a_j|$, from (18.6) we have

$$|\Delta A| \le r\gamma_{cm}' e(e^T|A|) =: mr\gamma_{cm}' G|A|,$$

where $||G||_F = 1$ (since $||ee^T||_F = m$). Finally, if $n = 1$, so that A is a column vector, then (as in the proof of Lemma 18.2) we can rewrite (18.5) as $\widehat{a}^{(r+1)} = (Q + \Delta Q)^T a$, where $\Delta Q^T = (Q^T \Delta a)a^T/a^T a$ and $\|\Delta Q\|_F = \|\Delta a\|_2/\|a\|_2 \le r\gamma_{cm}$. □

Note that the componentwise bound for *AA* in Lemma 18.3 does not imply the normwise one, because of the extra factor m in the componentwise bound. This is a nuisance, because it means we have to state both bounds in this and other analyses.

We now apply Lemma 18.3 to the computation of the *QR* factorization of a matrix.

Theorem 18.4. *Let* $\widehat{R} \in \mathbb{R}^{m\times n}$ *be the computed upper trapezoidal QR factor of* $A \in \mathbb{R}^{m\times n}$ *(*$m \ge n$*) obtained via the Householder QR algorithm. Then there exists an orthogonal* $Q \in \mathbb{R}^{m\times m}$ *such that*

$$A + \Delta A = Q\widehat{R},$$

where $||DA||_F \le ng_{cm}||A||_F$ *and* $|DA| \le mng_{cm}G|A|$*, with* $||G||_F = 1$*. The matrix Q is given explicitly as* $Q = (P_nP_{n-1} \ . \ . \ . \ P_1)^T$*, where* P_k *is the Householder matrix that corresponds to the exact application of the kth step of the algorithm to* A_k*.*

Proof. This is virtually a direct application of Lemma 18.3, with P_k defined as the Householder matrix that produces zeros below the diagonal in the *k*th column of the computed matrix A_k. one subtlety is that we do not explicitly compute the lower triangular elements of *R*, but rather set them to zero explicitly. However, it is easy to see that the conclusions of Lemmas 18.2 and 18.3 are still valid in these circumstances; the essential reason is that the elements of *DPb* in Lemma 18.2 that correspond to elements that are zeroed by the Householder matrix *P* are forced to be zero, and hence we can set the corresponding rows of *DP* to zero too, without compromising the bound on $||DP||_F$. □

Finally, we consider use of the *QR* factorization to solve a linear system. Given a *QR* factorization of a nonsingular matrix $A \in \mathbb{R}^{n\times n}$, a linear system $Ax = b$ can be solved by forming Q^Tb and then solving $Rx = Q^Tb$.

From Theorem 18.4, the computed $\widehat{R}$ is guaranteed to be nonsingular if $\kappa_2(A)n^{3/2}\gamma_{cm} < 1$. We give only componentwise bounds.

Theorem 18.5. *Let $A \in \mathbb{R}^{n\times n}$ be nonsingular. Suppose we solve the system $Ax = b$ with the aid of a QR factorization computed by the Householder algorithm. The computed $\widehat{x}$ satisfies*

$$(A + \Delta A)\widehat{x} = b + \Delta b,$$

where

$$|\Delta A| \le n^2\gamma_{cn}G|A|, \quad |\Delta b| \le n^2\gamma_{cn}G|b|, \qquad \|G\|_F = 1.$$

Proof. By A Theorem 18.4, the computed upper triangular factor $\widehat{R}$ satisfies $A + DA = QR$ with $|DA| \le n^2 g_{cn} G_1|A|$ and $\|G_1\|_F = 1$. BY Lemma 18.3, the computed transformed right-hand side satisfies $\widehat{c} = Q^T(b + \Delta b)$ with $|\Delta b| \le n^2\gamma_{cn}G_1|b|$. Importantly, the same orthogonal matrix Q appears in the equations involving $\widehat{R}$ and $\widehat{c}$.

By Theorem 8.5, the computed solution $\widehat{x}$ to the triangular system $\widehat{R}x = \widehat{c}$ satisfies

$$(\widehat{R} + \Delta R)\widehat{x} = \widehat{c}, \quad |\Delta R| \le \gamma_n|\widehat{R}|.$$

Premultiplying by Q yields

$$(A + \Delta A + Q\Delta R)\widehat{x} = b + \Delta b,$$

that is, $(A+\overline{\Delta A})\widehat{x} = b+\Delta b$, where $\overline{\Delta A} = \Delta A+Q\Delta R$. Using $\widehat{R} = Q^T(A+\Delta A)$ we have

$$\begin{aligned} |\overline{\Delta A}| &\le n^2\gamma_{cn}G_1|A| + \gamma_n|Q||\widehat{R}| \\ &\le n^2\gamma_{cn}G_1|A| + \gamma_n|Q||Q^T|(I + n^2\gamma_{cn}G_1)|A| \\ &=: n^2\gamma'_{cn}G|A|, \end{aligned}$$

where $G \ge G_1$ and $\|G\|_F = 1$. ▯

The proof of Theorem 18.5 naturally leads to a result in which b is perturbed. However, we can easily modify the proof so that only A is perturbed: the trick is to use Lemma 18.3 to write $\widehat{c} = (Q+\Delta Q)^T b$, where $\|\Delta Q\|_F \le n\gamma_{cn}$, and to premultiply by $(Q + DQ)^{-T}$ instead of Q in the middle of the proof. This leads to the result

$$(A + \Delta A)\widehat{x} = b, \quad |\Delta A| \le n^2\gamma_{cn}G|A|, \quad \|G\|_F = 1. \tag{18.7}$$

An interesting application of Theorem 18.5 is to iterative refinement, as explained in §18.6.

18.4. Aggregated Householder Transformations

In Chapter 12 we noted that LU factorization algorithms can be partitioned so as to express the bulk of the computation as matrix-matrix operations (level-3 BLAS). For computations with Householder transformations the same goal can be achieved by aggregating the transformations. This technique is widely used in LAPACK.

One form of aggregation is the "WY" representation of Bischof and Van Loan [105, 1987]. This involves representing the product $Q_r = P_r P_{r-1} \dots P_1$ of r Householder transformations $P_i = I - v_i v_i^T \in \mathbb{R}^{m\times m}$ (where $v_i^T v_i = 2$). in the form

$$Q_r = I + W_r Y_r^T, \qquad W_r, Y_r \in \mathbb{R}^{m\times r}.$$

This can be done using the recurrence

$$W_1 = -v_1, \;\; Y_1 = v_1, \qquad W_i = [W_{i-1} \;\; -v_i], \;\; Y_i = [Y_{i-1} \;\; Q_{i-1}^T v_i]. \tag{18.8}$$

Using the WY representation, a partitioned *QR* factorization can be developed as follows. Partition $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$ as

$$A = [\,A_1 \quad B\,], \qquad A_1 \in \mathbb{R}^{m\times r}, \tag{18.9}$$

and compute the Householder *QR* factorization of A_1,

$$P_r P_{r-1} \dots P_1 A_1 = \begin{bmatrix} R_1 \\ 0 \end{bmatrix}.$$

The product $P_r P_{r-1} \dots P_1 = I + W_r Y_r^T$ is accumulated using (18.8), as the P_i are generated, and then B is updated according to

$$B \leftarrow (I + W_r Y_r^T) B = B + W_r (Y_r^T B),$$

which involves only level-3 BLAS operations. The process is now repeated on the last $m - r$ rows of B.

When considering numerical stability, two aspects of the WY representation need investigating: its construction and its application. For the construction, we need to show that $\widehat{Q}_r := I + \widehat{W}_r \widehat{Y}_r^T$ satisfies

$$\|\widehat{Q}_r \widehat{Q}_r^T - I\|_2 \le d_1 u, \tag{18.10}$$

$$\|\widehat{W}_r\|_2 \le d_2, \quad \|\widehat{Y}_r\|_2 \le d_3, \tag{18.11}$$

for modest constants d_1, d_2, and d_3. Now

$$\widehat{Q}_i := I + [\,\widehat{W}_{i-1} \quad -\widehat{v}_i\,] \begin{bmatrix} \widehat{Y}_{i-1}^T \\ fl(\widehat{v}_i^T \widehat{Q}_{i-1}) \end{bmatrix} = \widehat{Q}_{i-1} - \widehat{v}_i fl(\widehat{v}_i^T \widehat{Q}_{i-1}).$$

But this last equation is essentially a standard multiplication by a Householder matrix, $\widehat{Q}_i = (I - \widehat{v}_i\widehat{v}_i^T)\widehat{Q}_{i-1}$, albeit with less opportunity for rounding errors. It follows from Lemma 18.3 that the near orthogonality of $\widehat{Q}_{i-1}$ is inherited by $\widehat{Q}_i$; the condition on $\widehat{Y}_r$ in (18.11) follows similarly and that on $\widehat{W}_r$ is trivial. Note that the condition (18. 10) implies that

$$\widehat{Q}_r = U_r + \Delta U_r, \quad U_r^T U_r = I, \quad \|\Delta U_r\|_2 \le d_1 u, \tag{18.12}$$

that is, $\widehat{Q}$ is close to an exactly orthogonal matrix (see Problem 18.13).

Next we consider the application of $\widehat{Q}_r$. Suppose we form $C = \widehat{Q}_r B = (I + \widehat{W}_r\widehat{Y}_r^T)B$ for the B in (18.9), so that

$$\widehat{C} = fl\big(B + fl(\widehat{W}_r(\widehat{Y}_r^T B))\big).$$

Analysing this level-3 BLAS-based computation using (18.12) and the very general assumption (12.3) on matrix multiplication (for the 2-norm), it is straightforward to show that

$$\begin{aligned} \widehat{C} &= U_r B + \Delta C = U_r(B + U_r^T \Delta C), \\ \|\Delta C\|_2 &\le \big[1 + d_1 + d_2 d_3(1 + c_1(r, m, n-r) \\ &\qquad + c_1(m, r, n-r))\big] u \|B\|_2 + O(u^2). \end{aligned} \tag{18.13}$$

This result shows that the computed update is an exact orthogonal update of a perturbation of B, where the norm of the perturbation is bounded in terms of the error constants for the level-3 BLAS.

Two conclusions can be drawn. First, algorithms that employ the WY representation with conventional level-3 BLAS are as stable as the corresponding point algorithms. Second, the use of fast BLAS3 for applying the updates affects stability only through the constants in the backward error bounds. The same conclusions apply to the more storage-efficient compact WY representation of Schreiber and Van Loan [905, 1989], and the variation of Puglisi [848, 1992].

18.5. Givens Rotations

Another way to compute the *QR* factorization is with Givens rotations. A Givens rotation (or plane rotation) $G(i, j, q) \in \mathbb{R}^{n \times n}$ is equal to the identity matrix except that

$$G([i, j], [i, j]) = \begin{bmatrix} c & s \\ -s & c \end{bmatrix},$$

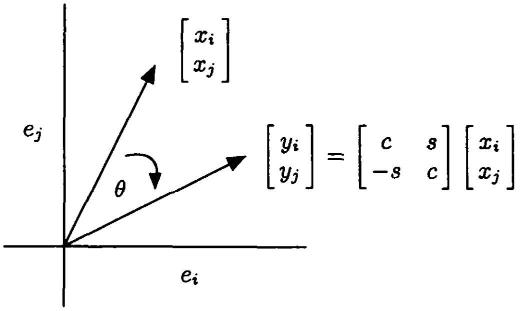

$\begin{bmatrix} x_i \\ x_j \end{bmatrix}$
e_j
θ
$\begin{bmatrix} y_i \\ y_j \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix}$
e_i

Figure 18.2. *Givens rotation, y = G(i, j, q)x.*

where $c = \cos q$ and $s = \sin q$. The multiplication $y = G(i,\ j,\ q)x$ rotates x through q radians clockwise in the $(i,\ j)$ plane; see Figure 18.2. Algebraically,

$$y_k = \begin{cases} x_k, & k \neq i, j, \\ cx_i + sx_j, & k = i, \\ -sx_i + cx_j, & k = j, \end{cases}$$

and so $y_j = 0$ if

$$s = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}, \qquad c = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}. \tag{18.14}$$

Givens rotations are therefore useful for introducing zeros into a vector one at a time. Note that there is no need to work out the angle q, since c and s in (18.14) are all that are needed to apply the rotation. In practice, we would scale the computation to avoid overflow (cf. §25.8).

To compute the *QR* factorization, Givens rotations are used to eliminate the elements below the diagonal in a systematic fashion. Various choices and orderings of rotations can be used; a natural one is illustrated as follows for a generic 4 x 3 matrix:

$$A = \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{G_{34}} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G_{23}} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G_{12}}$$

$$\left[\begin{array}{c|cc} \times & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{array}\right] \xrightarrow{G_{34}} \left[\begin{array}{c|cc} \times & \times & \times \\ \hline 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{array}\right] \xrightarrow{G_{23}} \left[\begin{array}{cc|c} \times & \times & \times \\ 0 & \times & \times \\ \hline 0 & 0 & \times \\ 0 & 0 & \times \end{array}\right] \xrightarrow{G_{34}}$$

$$\begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & 0 \end{bmatrix} = R.$$

The operation count for Givens *QR* factorization of a general m x n matrix ($m \geq n$) is $3n^2(m - n/3)$ flops, which is 50% more than that for Householder *QR* factorization. The main use of Givens rotations is to operate unstructured matrices—for example, to compute the *QR* factorization of a tridiagonal or Hessenberg matrix, or to carry out delicate zeroing in updating or downdating problems [470, 1989, §12.6].

Error analysis for Givens rotations is similar to that for Householder matrices—but a little easier. We omit the (straightforward) proof of the first result.

Lemma 18.6. *Let a Givens rotation G(i, j, q) be instructed according to* (18.14). *The computed* $\widehat{c}$ *and* $\widehat{s}$ *satisfy*

$$\widehat{c} = c(1 + \theta_4), \quad \widehat{s} = s(1 + \theta_4'), \tag{18.15}$$

where $|\theta_4|, |\theta_4'| \leq \gamma_4$. ☐

Lemma 18.7. *Let* $x \in \mathbb{R}^m$ *and consider the computation of* $y = \widehat{G}_{ij}x$, *where* $\widehat{G}_{ij}$ *is a computed Givens rotation in the (i, j) plane for which* $\widehat{c}$ *and* $\widehat{s}$ *satisfy* (18.15). *The computed* $\widehat{y}$ *satisfies*

$$\widehat{y} = (G_{ij} + \Delta G_{ij})x, \qquad \|\Delta G_{ij}\|_F \leq \sqrt{2}\gamma_6,$$

where G_{ij} *is an exact Givens rotation based on c and s in (18.15). All the rows of* DG_{ij} *except the ith and jth are zero.*

Proof. The vector $\widehat{y}$ differs from *x* only in elements *i* and *j*. We have

$$\widehat{y}_i = fl(\widehat{c}x_i + \widehat{s}x_j) = cx_i(1 + \theta_6) + sx_j(1 + \theta_6'),$$

where $|\theta_6|, |\theta_6'| \leq \gamma_6$, and similarly for $\widehat{y}_j$. Hence

$$|\widehat{y} - G_{ij}x| \leq \gamma_6 |G_{ij}||x|,$$

so that $\|\widehat{y} - G_{ij}x\|_2 \leq \sqrt{2}\gamma_6\|x\|_2$. We take $\Delta G_{ij} = (\widehat{y} - G_{ij}x)x^T/x^Tx$. ☐

For the next result we need the notion of disjoint Givens rotations. Rotations $G_{i1,j1}, \ldots, G_{ir,jr}$ are disjoint if $i_s \neq j_s$ and $j_s \neq j_t$ for $s \neq t$. Disjoint rotations are "nonconflicting" and therefore commute; it matters neither mathematically nor numerically in which order the rotations are applied. (Disjoint rotations can therefore be applied in parallel, though that is not our interest here.) Our approach is to take a given sequence of rotations and reorder them into groups of disjoint rotations. The reordered algorithm is numerically equivalent to the original one, but allows a simpler error analysis.

As an example of a rotation sequence already ordered into disjoint groups, consider the following sequence and ordering illustrated for a 6 × 5 matrix:

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ 1 & \times & \times & \times & \times \\ 2 & 3 & \times & \times & \times \\ 3 & 4 & 5 & \times & \times \\ 4 & 5 & 6 & 7 & \times \\ 5 & 6 & 7 & 8 & 9 \end{bmatrix}.$$

Here, an integer k in position (i, j) denotes that the (i, j) element is eliminated on the kth step by a rotation in the (j, i) plane, and all rotations on the kth step are disjoint. For an $m \times n$ matrix with $m > n$ there are $r = m + n$ — 2 stages, and the Givens QR factorization can be written as $W_r W_{r-1} \ . \ . \ . \ W_1 A = R$, where each W_i is a product of at most n disjoint rotations. It is easy to see that an analogous grouping into disjoint rotations can be done for the scheme illustrated at the start of this section.

Lemma 18.8. *Consider the sequence of transformations*

$$A_{k+1} = W_k A_k, \qquad k = 1{:}r,$$

where $A_1 = A \in \mathbb{R}^{m\times n}$ *and each* W_k *is a product of disjoint Givens rotations. Assume that the individual Givens rotations are performed using computed sine and cosine values related to the exact values defining the* W_k *by* (18.15). *Then the computed matrix* $\hat{A}_{r+1}$ *satisfies*

$$\hat{A}_{r+1} = Q^T(A + DA),$$

where $Q^T = W_r W_{r-1} \ . \ . \ . \ W_1$ *and DA satisfies the normwise and componentwise bounds*

$$\begin{aligned} \|\Delta A\|_F &\le \gamma_{cr}\|A\|_F, \\ |\Delta A| &\le m\gamma_{cr}G|A|, \quad \|G\|_F = 1. \end{aligned}$$

(In fact, we can take $G = m^{-1}ee^T$, *where* $e = [1, 1, . . . , 1]^T$.*) In the special case* $n = 1$, *so that* $A = a$, *we have* $\hat{a}^{(r+1)} = (Q + DQ)^T a$ *with* $\|DQ\|_F \le g_{cr}$.

Proof. The proof is analogous to that of Lemma 18.3, so we offer only a sketch. First, we consider the jth column of A, a_j, which undergoes the transformations $a_j^{(r+1)} = W_r \ldots W_1 a_j$. By Lemma 18.7 and the disjointness of the rotations, we have

$$\widehat{a}_j^{(r+1)} = (W_r + \Delta W_r) \ldots (W_1 + \Delta W_1) a_j,$$

where each DW_k depends on j and satisfies $\|DW_k\|_2 \le \sqrt{2}\gamma_6$. Using Lemma 3.6 we obtain

$$\begin{aligned} \widehat{a}_j^{(r+1)} &= Q^T(a_j + d_j), \\ \|d_j\|_2 &\le \big((1+\sqrt{2}\gamma_6)^r - 1\big)\|a_j\|_2 = \gamma_{cr}\|a_j\|_2. \end{aligned} \tag{18.16}$$

Hence

$$\|\Delta A\|_F^2 = \sum_{j=1}^{n} \|d_j\|_2^2 \le \gamma_{cr}^2 \|A\|_F^2.$$

The inequalities (18. 16) for $j = 1{:}n$ imply that

$$|\Delta A| \le \gamma_{cr} e(e^T|A|) =: m\gamma_{cr} G|A|,$$

where $\|G\|_F = 1$. The result for $n = 1$ is proved as in Lemma 18.3. II

We are now suitably equipped to give a result for Givens QR factorization.

Theorem 18.9. *Let $\widehat{R} \in \mathbb{R}^{m\times n}$ be the computed upper trapezoidal QR factor of $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$ obtained via the Givens QR algorithm, with any standard choice and ordering of rotations. Then there exists an orthogonal $Q \in \mathbb{R}^{m\times m}$ such that*

$$A + \Delta A = Q\widehat{R},$$

with $\|DA\|_F \le g_{c(m+n)}\|A\|_F$ and $|DA| \le mg_{c(m+n)}G|A|$, $\|G\|_F = 1$. (The matrix Q is a product of Givens rotations, the kth of which corresponds to the exact application of the kth step of the algorithm to $\hat{A}_k$.) □

It is interesting that the error bounds for QR factorization with Givens rotations are a factor n smaller than those for Householder QR factorization. This appears to be an artefact of the analysis, and we are not aware of any difference in accuracy in practice.

18.6. Iterative Refinement

Consider a nonsingular linear system $Ax = b$, where $A \in \mathbb{R}^{n\times n}$. Suppose we solve the system using a QR factorization $A = QR$ computed using Householder or Givens transformations (thus, x is obtained by solving $Rx = Q^Tb$).

Theorem 18.5, and its obvious analogue for Givens rotations, show that $\widehat{x}$ satisfies

$$|b - A\widehat{x}| \le p(n)uG(|A||\widehat{x}| + |b|), \tag{18.17}$$

where $\|G\|_F = 1$ and p is a low-degree polynomial. Hence $\widehat{x}$ has a normwise backward error of order $p(n)u$. However, since G is a full matrix, (18.17) suggests that the componentwise relative backward error $\omega_{|A|,|b|}(\widehat{x})$ need not be small. In fact, we know of no nontrivial class of matrices for which Householder or Givens *QR* factorization is guaranteed to yield a small componentwise relative backward error.

Suppose that we carry out a step of fixed precision iterative refinement, to obtain $\widehat{y}$. The form of the bound (18.17) enables us to invoke Theorem 11.4. We conclude that the componentwise relative backward error $\omega_{|A|,|b|}(\widehat{y})$ after one step of iterative refinement will be small as long as A *is* not too ill conditioned and $|A||\widehat{y}|$ is not too badly scaled. This conclusion is similar to that for Gaussian elimination with partial pivoting (GEPP), except that for GEPP there is the added requirement that the LU factorization not suffer large element growth.

Recall from (18.7) that we also have a backward error result in which only A is perturbed. The analysis of $11.1 is therefore applicable, and analogues of Theorems 11.1 and 11.2 hold in which $h = u\kappa_\infty(A)$.

The performance of *QR* factorization with fixed precision iterative refinement is illustrated in Tables 11. 1–1 1.3 in §11.2. The performance is as predicted by the analysis. Notice that the initial componentwise relative backward error is large in Table 11.2 but that iterative refinement successfully reduces it to the roundoff level (despite $\mathrm{cond}(A^{-1})\sigma(A,x)$ being huge). It is worth stressing that the *QR* factorization yielded a small normwise relative backward error in each example ($\eta_{A,b}(\widehat{x}) < u$, in fact), as we know it must.

18.7. Gram–Schmidt Orthogonalization

The oldest method for computing a *QR* factorization is the Gram-Schmidt orthogonalization method. It can be derived directly from the equation $A = QR$, where $A, Q \in \mathbb{R}^{m\times n}$ and $R \in \mathbb{R}^{n\times n}$ (Gram-Schmidt does not compute the m x m matrix Q in the full *QR* factorization and hence does not provide a basis for the orthogonal complement of range(A)). Denoting by a_j and q_j the jth columns of A and Q, respectively, we have

$$a_j = \sum_{k=1}^{j} r_{kj} q_k.$$

Premultiplying by q_i^T yields, since Q has orthonormal columns, $q_i^T a_j = r_{ij}$, $i = 1:j-1$. Further,

$$q_j = q_j'/r_{jj},$$

where

$$q_j' = a_j - \sum_{k=1}^{j-1} r_{kj} q_k, \quad r_{jj} = \|q_j'\|_2.$$

Hence we can compute Q and R a column at a time. To ensure that $r_{jj} > 0$ we require that A has full rank.

Algorithm 18.10 (classical Gram–Schmidt). Given $A \in \mathbb{R}^{m\times n}$ of rank n this algorithm computes the QR factorization $A = QR$, where Q is $m \times n$ and R is $n \times n$, by the Gram-Schmidt method.

```
for $j = 1:n$
    for $i = 1:j-1$
        $r_{ij} = q_i^T a_j$
    end
    $q_j' = a_j - \sum_{k=1}^{j-1} r_{kj} q_k$
    $r_{jj} = \|q_j'\|_2$
    $q_j = q_j'/r_{jj}$
end
```

Cost: $2mn^2$ flops ($2n^3/3$ flops more than Householder QR factorization with Q left in factored form).

In the classical Gram–Schmidt method (CGS), a_j appears in the computation only at the jth stage. The method can be rearranged so that as soon as q_j is computed, all the remaining vectors are orthogonalized against q_j. This gives the modified Gram-Schmidt method (MGS).

Algorithm 18.11 (modified Gram-Schmidt). Given $A \in \mathbb{R}^{m\times n}$ of rank n this algorithm computes the QR factorization $A = QR$, where Q is $m \times n$ and R is $n \times n$, by the MGS method.

```
$a_k^{(1)} = a_k$, $k = 1:n$
for $k = 1:n$
    $r_{kk} = \|a_k^{(k)}\|_2$
    $q_k = a_k^{(k)}/r_{kk}$
    for $j = k+1:n$
        $r_{kj} = q_k^T a_j^{(k)}$
        $a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k$
    end
end
```

Cost: $2mn^2$ flops.

It is worth noting that there are two differences between the CGS and MGS methods. The first is the order in which the calculations are performed: in the modified method each remaining vector is updated once on each step instead of having all its updates done together on one step. This is purely a matter of the order in which the operations are performed. Second, and more crucially in finite precision computation, two different (but mathematically equivalent) formulae for r_{kj} are used: in the classical method, $r_{kj} = q_k^T a_j$, which involves the original vector a_j, whereas in the modified method a_j is replaced in this formula by the partially orthogonalized vector $a_j^{(k)}$. Another way to view the difference between the two Gram–Schmidt methods is via representations of an orthogonal projection; see Problem 18.7.

The MGS procedure can be expressed in matrix terms by defining $A_k = [q_1,\dots,q_{k-1},a_k^{(k)},\dots,a_n^{(k)}]$. MGS transforms $A_1 = A$ into $A_{n+1} = Q$ by the sequence of transformations $A_k = A_{k+1}R_k$, where R_k is equal to the identity except in the kth row, where it agrees with the final R. For example, if $n = 4$ and $k = 3$,

$$A_3 = \begin{bmatrix} q_1 & q_2 & a_3^{(3)} & a_4^{(3)} \end{bmatrix} = \begin{bmatrix} q_1 & q_2 & q_3 & a_4^{(4)} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & r_{33} & r_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} = A_4 R_3.$$

Thus $R = R_n \dots R_1$.

The Gram-Schmidt methods produce Q explicitly, unlike the Householder and Givens methods, which hold Q in factored form. While this is a benefit, in that no extra work is required to form Q, it is also a weakness, because there is nothing in the methods to force the computed $\widehat{Q}$ to be orthonormal in the face of roundoff. Orthonormality of Q is a consequence of orthogonality relations that are implicit in the methods, and these relations may be vitiated by rounding errors.

Some insight is provided by the case $n = 2$, for which the CGS and MGS methods are identical. Given $a_1, a_2 \in \mathbb{R}^m$ we compute $q_1 = a_1/\|a_1\|_2$, which we will suppose is done exactly, and then we form the unnormalized vector $q_2 = a_2 - (q_1^T a_2)q_1$. The computed vector satisfies

$$\widehat{q}_2 = a_2 - q_1^T(a_2 + \Delta a_2) + \Delta\widetilde{q}_2,$$

where

$$|\Delta a_2| \le \gamma_m |a_2|, \quad |\Delta\widetilde{q}_2| \le u|a_2 - q_1^T(a_2 + \Delta a_2)|.$$

Hence

$$\widehat{q}_2 = q_2 + \Delta q_2, \quad |\Delta q_2| \le \gamma_m |q_1^T||a_2| + u(|a_2| + (1+\gamma_m)|q_1^T||a_2|),$$

and so the normalized inner product satisfies

$$\left| q_1^T \frac{\widehat{q}_2}{\|\widehat{q}_2\|_2} \right| \lesssim (m+2)u \frac{\|a_2\|_2}{\|q_2\|_2} = \frac{(m+2)u}{\sin \angle(a_1, a_2)}, \tag{18.18}$$

where $\angle(a_1, a_2)$ is the angle between a_1 and a_2. But $\kappa_2(A) \geq \cot \angle(a_1, a_2)$, where $A = [a_1, a_2]$ (Problem 18.8). Hence, for $n = 2$, the loss of orthogonality can be bounded in terms of $\kappa_2(A)$. The same is true in general for the MGS method, as proved by Björck [107, 1967]. A direct proof is quite long and complicated, but a recent approach of Björck and Paige [119, 1992] enables a much shorter derivation; we take this approach here.

The observation that simplifies the error analysis of the MGS method is that the method is equivalent, both mathematically *and numerically,* to Householder *QR* factorization of the padded matrix $\left[\begin{smallmatrix} 0_n \\ A \end{smallmatrix}\right] \in \mathbb{R}^{(m+n)\times n}$. To understand this equivalence, consider the Householder *QR* factorization

$$P^T \begin{bmatrix} 0_n \\ A \end{bmatrix} = \begin{bmatrix} R \\ 0 \end{bmatrix}, \qquad P^T = P_n \dots P_2 P_1. \tag{18.19}$$

Let $q_1, \dots, q_n \in \mathbb{R}^m$ be the vectors obtained by applying the MGS method to A. Then it is easy to see that

$$P_1 = I - v_1 v_1^T, \quad v_1 = \begin{bmatrix} -e_1 \\ q_1 \end{bmatrix}, \quad v_1^T v_1 = 2$$

and that the multiplication $A_2 = P_1 \left[\begin{smallmatrix} 0_n \\ A \end{smallmatrix}\right]$ carries out the first step of the MGS method on A, producing the first row of R and $a_2^{(2)}, \dots, a_n^{(2)}$:

$$A_2 = \begin{array}{c} 1 \\ n-1 \\ m \end{array} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & 0 & \dots & 0 \\ 0 & a_2^{(2)} & \dots & a_n^{(2)} \end{bmatrix}.$$

The argument continues in the same way, and we find that

$$P_k = I - v_k v_k^T, \quad v_k = \begin{bmatrix} -e_k \\ q_k \end{bmatrix}, \quad v_k^T v_k = 2, \quad k = 2\colon n. \tag{18.20}$$

With the Householder–MGS connection established, we are ready to derive error bounds for the MGS method by making use of our existing error analysis for the Householder method.

Theorem 18.12. *Suppose the MGS method is applied to $A \in \mathbb{R}^{m\times n}$ of rank n, yielding computed matrices $\widehat{Q} \in \mathbb{R}^{m\times n}$ and $\widehat{R} \in \mathbb{R}^{n\times n}$. Then there are constants $c_i = c_i(m, n)$ such that*

$$A + \Delta A_1 = \widehat{Q}\widehat{R}, \qquad \|\Delta A_1\|_2 \leq c_1 u \|A\|_2, \tag{18.21}$$

$$\|\widehat{Q}^T \widehat{Q} - I\|_2 \leq c_2 u \kappa_2(A) + O\big((u\kappa_2(A))^2\big), \tag{18.22}$$

and there exists an orthonormal matrix Q such that

$$A + \Delta A_2 = Q\widehat{R}, \qquad |\Delta A_2| \le c_3 u G|A|, \quad \|G\|_F = 1. \tag{18.23}$$

Proof. To prove (18.21) we use the matrix form of the MGS method. For the computed matrices we have

$$\widehat{A}_k = \widehat{A}_{k+1}\widehat{R}_k + \Delta_k, \qquad |\Delta_k| \le u|\widehat{A}_{k+1}||\widehat{R}_k|.$$

Expanding this recurrence, we obtain

$$A = \widehat{Q}\widehat{R} + \Delta_n\widehat{R}_{n-1}\dots\widehat{R}_1 + \Delta_{n-1}\widehat{R}_{n-2}\dots\widehat{R}_1 + \cdots + \Delta_2\widehat{R}_1 + \Delta_1.$$

Hence

$$|A - \widehat{Q}\widehat{R}| \le u\big(|\widehat{A}_{n+1}||\widehat{R}_n|\dots|\widehat{R}_1| + \cdots + |\widehat{A}_3||\widehat{R}_2||\widehat{R}_1| + |\widehat{A}_2||\widehat{R}_1|\big), \tag{18.24}$$

and a typical term has the form

$$|\widehat{A}_k||\widehat{R}_{k-1}|\dots|\widehat{R}_1| = \big|[\widehat{q}_1 \ \dots \ \widehat{q}_{k-1} \ \widehat{a}_k^{(k)} \ \dots \ \widehat{a}_n^{(k)}]\big|S_{k-1}, \tag{18.25}$$

where S_{k-1} agrees with $|\widehat{R}|$ in its first $k-1$ rows and the identity in its last $n-k+1$ rows. Assume for simplicity that $\|\widehat{q}_i\|_2 \equiv 1$ (this does not affect the final result). We have $a_j^{(k+1)} = (I - q_k q_k^T)a_j^{(k)}$, and Lemma 3.8 shows that the computed vector satisfies

$$\|\widehat{a}_j^{(k+1)}\|_2 \le (1 + 2\gamma_{m+3})\|\widehat{a}_j^{(k)}\|_2,$$

which implies $\|\widehat{A}_{k+1}\|_F \le (1 + 2\gamma_{m+3})^k\|A\|_F$ and, from $\widehat{r}_{kj} = fl(\widehat{q}_k^T\widehat{a}_j^{(k)})$, we have $\|\widehat{R}\|_F \le \sqrt{n}(1+\gamma_m)(1+2\gamma_{m+3})^{n-1}\|A\|_F$. Using (18.24) and exploiting the form of (18.25) we find, after a little working, that

$$\|A - \widehat{Q}\widehat{R}\|_F \le 4n^2u\|A\|_F,$$

provided that $(1+\gamma_m)(1+2\gamma_{m+3})^{n-1} < 2$.

To prove the last two parts of the theorem we exploit the Householder–MGS connection. By applying Theorem 18.4 to (18.19) we find that there is an orthogonal $\widetilde{P}$ such that

$$\begin{bmatrix} \Delta A_3 \\ A + \Delta A_4 \end{bmatrix} = \widetilde{P}\begin{bmatrix} \widehat{R} \\ 0 \end{bmatrix} = \begin{bmatrix} \widetilde{P}_{11} \\ \widetilde{P}_{21} \end{bmatrix}\widehat{R}, \tag{18.26}$$

with

$$|\Delta A_i| \le c_4 u G_i|A|, \quad \|G_i\|_F = 1, \quad i = 3\colon 4.$$

This does not directly yield (18.23), since $\widetilde{P}_{21}$ is not orthonormal. However, it can be shown that if we define Q to be the nearest orthonormal matrix to $\widetilde{P}_{21}$ in the Frobenius norm, then (18.23) holds with $c_3 = \sqrt{m} + 1)c_4$ (see Problem 18.11).

Now (18.21) and (18.23) yield $\widehat{Q} - Q = (\Delta A_1 - \Delta A_2)\widehat{R}^{-1}$, so

$$\|\widehat{Q} - Q\|_2 \le (c_1 + \sqrt{n}c_3)u\|A\|_2\|\widehat{R}^{-1}\|_2 \le \frac{c_5 u\kappa_2(A)}{1 - \sqrt{n}c_3 u\kappa_2(A)},$$

where $c_5 = c_1 + \sqrt{n}c_3$ and we have used (18.23) to bound $\|\widehat{R}^{-1}\|_2$. This bound implies (18.22) with $c_2 = 2c_5$ (use the first inequality in Problem 18.13). $\square$

We note that (18.22) can be strengthened by replacing $\kappa_2(A)$ in the bound by the minimum over positive diagonal matrices D of $\kappa_2(AD)$. This follows from the observation that in the MGS method the computed Q is invariant under scalings $A \leftarrow AD$, at least if D comprises powers of the machine base. As a check, note that the bound in (18.18) for the case $n = 2$ is independent of the column scaling, since $\sin \angle(a_1, a_2)$ is.

Theorem 18.12 tells us three things. First, the computed *QR* factors from the MGS method have a small residual. Second, the departure from orthonormality of Q is bounded by a multiple of $\kappa_2(A)u$, so that Q is guaranteed to be nearly orthonormal if A is well conditioned. Finally, R is the exact triangular *QR* factor of a matrix near to A in a componentwise sense, so it is as good an R-factor as that produced by Householder *QR* factorization applied to A. In terms of the error analysis, the MGS method is weaker than Householder *QR* factorization only in that Q is not guaranteed to be nearly orthonormal.

For the CGS method the residual bound (18.21) still holds, but no bound of the form (18.22) holds for $n > 2$ (see Problem 18.9).

Here is a numerical example to illustrate the behaviour of the Gram–Schmidt methods. We take the 25×15 Vandermonde matrix $A = (p_i^{j-1})$, where the p_i are equally spaced on $[0, 1]$. The condition number $\kappa_2(A) = 1.47 \times 10^9$. We have

$$\begin{aligned} \text{CGS}: &\quad \|A - \widehat{Q}\widehat{R}\|_2 = 5.0 \times 10^{-16}, \quad \|\widehat{Q}^T\widehat{Q} - I\|_2 = 3.9, \\ \text{MGS}: &\quad \|A - \widehat{Q}\widehat{R}\|_2 = 3.0 \times 10^{-16}, \quad \|\widehat{Q}^T\widehat{Q} - I\|_2 = 6.2 \times 10^{-9}. \end{aligned}$$

Both methods produce a small residual for the *QR* factorization. While CGS produces a Q showing no semblance of orthogonality, for MGS we have $\|\widehat{Q}^T\widehat{Q} - I\|_2 \approx \kappa_2(A)u/26$.

18.8. Sensitivity of the *QR* Factorization

How do the *QR* factors of a matrix behave under small perturbations of the matrix? This question was first considered by Stewart [944, 1977]. He showed that if $A \in \mathbb{R}^{m\times n}$ has rank *n* and

$$A = QR \quad \text{and} \quad A + \Delta A = (Q + \Delta Q)(R + \Delta R)$$

are *QR* factorization, then, for sufficiently small *DA*,

$$\frac{\|\Delta R\|_F}{\|R\|_F} \le c_n \kappa_F(A) \frac{\|\Delta A\|_F}{\|A\|_F}, \tag{18.27}$$

$$\|\Delta Q\|_F \le c_n \kappa_F(A) \frac{\|\Delta A\|_F}{\|A\|_F}, \tag{18.28}$$

where c_n is a constant. Here, and throughout this section, we use the "economy size" *QR* factorization with *R* a square matrix normalized to have nonnegative diagonal elements. Similar normwise bounds are given by Stewart [951, 1993] and Sun [971, 1991], and, for *AQ* only, by Bhatia and Mukherjea [95, 1994] and Sun [975, 1995].

Componentwise sensitivity analyses have been given by Zha [1126, 1993] and Sun [972, 1992], [973, 1992]. Zha's bounds can be summarized as follows, with the same assumptions and notation as for Stewart's result above. Let $|DA| \le \epsilon G|A|$, where *G* is nonnegative with $\|G\|_\infty = 1$. Then, for sufficiently small ϵ,

$$\begin{aligned} \frac{\|\Delta R\|_\infty}{\|R\|_\infty} &\le c_{m,n}\epsilon \operatorname{cond}(R^{-1}) + O(\epsilon^2), \\ \|\Delta Q\|_\infty &\le c_{m,n}\epsilon \operatorname{cond}(R^{-1}) + O(\epsilon^2), \end{aligned} \tag{18.29}$$

where $c_{m,n}$ is a constant depending on *m* and *n*. The quantity $f(A) = \operatorname{cond}(R^{-1})$ can therefore be thought of as a condition number for the *QR* factorization under the columnwise class of perturbations considered. Note that *f* is independent of the column scaling of *A*.

As application of these bounds, consider a computed *QR* factorization $A \approx QR$ obtained via the Householder algorithm, where *Q* is the computed product of the computed Householder matrices. Theorem 18.4 shows that $A + DA = \widetilde{Q}\widehat{R}$, where $\widetilde{Q}$ is orthogonal and $|\Delta A| \le mn\gamma_{cm} G_1 |A|$, with $\|G_1\|_F = 1$. Applying Lemma 18.3 to_ the computation of $\widehat{Q}$ we have that $\widehat{Q} = \widetilde{Q} + \Delta$, where $|\Delta| \le mn\gamma_{cm} G_2^T |\widetilde{Q}|$, with $\|G_2\|_F = 1$. From the expression $Q - \widehat{Q} = (Q - \widetilde{Q}) + (\widetilde{Q} - \widehat{Q}) = (Q - \widetilde{Q}) - \Delta$ we have, on applying (18.29) to the first term,

$$\|Q - \widehat{Q}\|_\infty \le c_{m,n} u \phi(A) + O(u^2). \tag{18.30}$$

To illustrate this analysis we describe an example of Zha. Let

$$A = \begin{bmatrix} 1 & 1 \\ 0 & \epsilon \\ 0 & 1 \end{bmatrix}, \qquad B = \begin{bmatrix} 1 & 1/\epsilon \\ 0 & 1/\epsilon \\ 0 & 1/\epsilon \end{bmatrix},$$

where $\epsilon > 0$ is a parameter. It is easy to verify that $A = QR_A$ and $B = QR_B$ are QR factorization (the same Q in each case) where

$$Q = -\begin{bmatrix} 1/\sqrt{2} & 0 \\ 0 & 1 \\ 1/\sqrt{2} & 0 \end{bmatrix}.$$

With $\epsilon = 10^{-8}$, the computed Q factors from MATLAB are

```
Q_A =                                Q_B =
 -7.0711e-001  1.2539e-008           -7.0711e-001  1.5872e-016
            0 -1.0000e+000                      0 -1.0000e+000
 -7.0711e-001 -1.2539e-008           -7.0711e-001 -1.5872e-016
```

and

$$\|Q - Q_A\|_2 \approx 1.8 \times 10^{-8}, \qquad \|Q - Q_B\|_2 \approx 2.2 \times 10^{-16}.$$

Since $f(A) \approx 2.8 \times 10^8$ and $f(B) \approx 3.8$, the actual errors match the bound (18.30) very well. Note that $\kappa_2(A) \approx 2.8 \times 10^8$ and $\kappa_2(B) \approx 2.1 \times 10^8$, so the normwise perturbation bound (18.28) is not strong enough to predict the difference in accuracy of the computed Q factors, unless we scale the factor $1/\epsilon$ out of the last column of B to leave a well-conditioned matrix. The virtue of the componentwise analysis is that it does not require a judicious scaling in order to yield useful results.

18.9. Notes and References

The earliest appearance of Householder matrices is in the book by Turnbull and Aitken [1029, 1932, pp. 102–105]. These authors show that if $\|x\|_2 = \|y\|_2$ $(x \neq -y)$ then a unitary matrix of the form $R = azz^* - I$ (in their notation) can be constructed so that $Rx = y$. They use this result to prove the existence of the Schur decomposition. The first systematic use of Householder matrices for computational purposes was by Householder [586, 1958], who used them to construct the QR factorization. Householder's motivation was to compute the QR factorization with less arithmetic operations (in particular, less square roots) than are required by the use of Givens rotations.

In the construction of §18.1 for a Householder matrix P such that $Px = se_1$, the other choice of sign, $\mathrm{sign}(s) = \mathrm{sign}(x_1)$, can be used, provided that v_1 is formed in a numerically stable way. The appropriate formula is derived as follows [264, 1976], [819, 1971], [820, 1980, p. 91]:

$$v_1 = x_1 - \mathrm{sign}(x_1)\|x\|_2 = \frac{x_1^2 - \|x\|_2^2}{x_1 + \mathrm{sign}(x_1)\|x\|_2} = \frac{-(x_2^2 + \cdots + x_n^2)}{x_1 + \mathrm{sign}(x_1)\|x\|_2}.$$

A detailed analysis of different algorithms for constructing P such that $Px = se_1$ is given by Parlett [819, 1971].

Tsao [1023, 1975] describes an alternative way to form the product of a Householder matrix with a vector and gives an error analysis. There is no major advantage over the usual approach.

As for Householder matrices, normwise error analysis for Givens rotations was given by Wilkinson [1087, 1963], [1089, 1965, pp. 131–139]. Wilkinson analysed *QR* factorization by Givens rotations for square matrices [1089, 1965, pp. 240-241], and his analysis was extended to rectangular matrices by Gentleman [435, 1973]. The idea of exploiting disjoint rotations in the error analysis was developed by Gentleman [436, 1975], who gave a normwise analysis that is simpler and produces smaller bounds than Wilkinson's (our normwise bound in Theorem 18.9 is essentially the same as Gentleman's).

For more details of algorithmic and other aspects of Householder and Givens *QR* factorization, see Golub and Van Loan [470, 1989, §5.2].

The error analysis in §18.3 is a refined and improved version of analysis that appeared in the technical report [546, 1990] and was quoted without proof in Higham [549, 1991].

The WY representation for a product of Householder transformations should not be confused with a genuine block Householder transformation. Schreiber and Parlett [904, 1988] define, for a given $Z \in \mathbb{R}^{m\times n}$ $(m \geq n)$, the "block reflector that reverses the range of Z" as

$$H = I_m - ZWZ^T, \qquad W = 2(Z^TZ)^+ \in \mathbb{R}^{n\times n}.$$

If $n = 1$ this is just a standard Householder transformation. A basic task is as follows: given $E \in \mathbb{R}^{m\times n}$ $(m > n)$ find a block reflector H such that

$$HE = \begin{bmatrix} F \\ 0 \end{bmatrix}, \qquad F \in \mathbb{R}^{n\times n}.$$

Schreiber and Parlett develop theory and algorithms for block reflectors, in both of which the polar decomposition plays a key role.

Sun and Bischof [977, 1995] show that any orthogonal matrix can be expressed in the form $Q = I - YSY^T$, even with S triangular, and they explore the properties of this representation.

Another important use of Householder matrices, besides computation of the *QR* factorization, is to reduce a matrix to a simpler form prior to iterative computation of eigenvalues (Hessenberg or tridiagonal form) or singular values (bidiagonal form). For these two-sided transformations an analogue of Lemma 18.3 holds with normwise bounds (only) on the perturbation. Error analyses of two-sided application of Householder transformations is given by Ortega [811, 1963] and Wilkinson [1086, 1962], [1089, 1965, Chap. 6].

Mixed precision iterative refinement for solution of linear systems by Householder *QR* factorization is discussed by Wilkinson [1090, 1965, §10], who notes that convergence is obtained as long as a condition of the form $c_n\kappa_2(A)u < 1$ holds.

Fast Givens rotations can be applied to a matrix with half the number of multiplications of conventional Givens rotations, and they do not involve square roots. They were developed by Gentleman [435, 1973] and Hammarling [498, 1974]. Fast Givens rotations are as stable as conventional ones-see the error analysis by Parlett in [820, 1980, §6.8.3], for example-but, for the original formulations, careful monitoring is required to avoid overflow. Rath [861, 1982] investigates the use of fast Givens rotations for performing similarity transformations in solving the eigenproblem. Barlow and Ipsen [65, 1987] propose a class of scaled Givens rotations suitable for implementation on systolic arrays, and they give a detailed error analysis. Anda and Park [16, 1994] develop fast rotation algorithms that use dynamic scaling to avoid overflow.

Rice [870, 1966] was the first to point out that the MGS method produces a more nearly orthonormal matrix than the CGS method in the presence of rounding errors. Björck [107, 1967] gives a detailed error analysis, proving (18.21) and (18.22) but not (18.23), which is an extension of the corresponding normwise result of Björck and Paige [119, 1992]. Björck and Paige give a detailed assessment of MGS versus Householder QR factorization.

The difference between the CGS and MGS methods is indeed subtle. Wilkinson [1095, 1971] admitted that "I used the modified process for many years without even noticing explicitly that I was not performing the classical algorithm."

The orthonormality of the matrix $\widehat{Q}$ from Gram-Schmidt can be improved by reorthogonalization, in which the orthogonalization step of the classical or modified method is iterated. Analyses of Gram–Schmidt with reorthogonalization are given by Abdelmalek [2, 1971], Ruhe [883, 1983], and Hoffmann [578, 1989]. Daniel, Gragg, Kaufman, and Stewart [263, 1976] analyse the use of classical Gram–Schmidt with reorthogonalization for updating a QR factorization after a rank one change to the matrix.

The mathematical and numerical equivalence of the MGS method with Householder QR factorization of the matrix $\left[\begin{smallmatrix} 0_n \\ A \end{smallmatrix}\right]$ was known in the 1960s (see the Björck quotation at the start of the chapter) and the mathematical equivalence was pointed out by Lawson and Hanson [695, 1974, Ex. 19.39].

A block Gram-Schmidt method is developed by Jalby and Philippe [608, 1991] and error analysis given. See also Björck [115, 1994], who gives an up-to-date survey of numerical aspects of the Gram–Schmidt method.

For more on Gram-Schmidt methods, including historical comments, see Björck [116, 1996].

One use of the QR factorization is to orthogonalize a matrix that, because of rounding or truncation errors, has lost its orthogonality; thus we compute $A = QR$ and replace A by Q. An alternative approach is to replace $A \in \mathbb{R}^{m\times n}$ ($m \ge n$) by the nearest orthonormal matrix, that is, the matrix Q that solves $\{\, \|A - Q\| : Q^TQ = I\} = \min$. For the 2- and Frobenius norms, the optimal

Q is the orthonormal polar factor U of A, where $A = UH$ is a polar decomposition: $U \in \mathbb{R}^{m\times n}$ has orthonormal columns and $H \in \mathbb{R}^{n\times n}$ is symmetric positive semidefinite. If $m = n$, U is the nearest orthogonal matrix to A in any unitarily invariant norm, as shown by Fan and Hoffman [361, 1955]. Chandrasekaran and Ipsen [196, 1994] show that the QR and polar factors satisfy $\|A - Q\|_{2,F} \le 5\sqrt{n}\|A - U\|_2$, under the assumptions that A has full rank and columns of unit 2-norm and that R has positive diagonal elements. Sun [974, 1995] proves a similar result and also obtains a bound for $\|Q - U\|_F$ in terms of $\|A^TA - I\|_F$. Algorithms for maintaining orthogonality in long products of orthogonal matrices, which arise, for example, in subspace tracking problems in signal processing, are analysed by Edelman and Stewart [347, 1993] and Mathias [733, 1995].

Various iterative methods are available for computing the orthonormal polar factor U, and they can be competitive in cost with computation of a QR factorization. For more details on the theory and numerical methods, see Higham [530, 1986], [539, 1989], Higham and Papadimitriou [567, 1994], and the references therein.

A notable omission from this chapter is a treatment of rank-revealing QR factorizations-ones in which the rank of A can readily be determined from R. This topic is not one where rounding errors play a major role, and hence it is outside the scope of this book. Pointers to the literature include Golub and Van Loan [470, 1989, §5.4], Chan and Hansen [192, 1992], and Björck [116, 1996]. A column pivoting strategy for the QR factorization, described in Problem 18.5, ensures that if A has rank r then only the first r rows of R are nonzero. A perturbation theorem for the QR factorization with column pivoting is given by Higham [540, 1990]; it is closely related to the perturbation theory in §10.3.1 for the Cholesky factorization of a positive semidefinite matrix.

18.9.1. LAPACK

LAPACK contains a rich selection of routines for computing and manipulating the QR factorization and its variants. Routine `xGEQRF` computes the QR factorization $A = QR$ of an $m \times n$ matrix A by the Householder QR algorithm. If $m < n$ (which we ruled out in our analysis, merely to simplify the notation), the factorization takes the form $A = Q[R_1 \ \ R_2]$, where R_1 is $m \times m$ upper triangular. The matrix Q is represented as a product of Householder transformations and is not formed explicitly. A routine `xORGQR` (or `xUNGQR` in the complex case) is provided to form all or part of Q, and routine `xORMQR` (or `xUNMQR`) will pre- or postmultiply a matrix by Q or its (conjugate) transpose.

Routine `xGEQPF` computes the QR factorization with column pivoting (see Problem 18.5).

An LQ factorization is computed by `xGELQF`. When A is $m \times n$ with $m \le n$

it takes the form $A = [L\ 0]\,Q$. It is essentially the same as a QR factorization of A^T and hence can be used to find the minimum 2-norm solution to an underdetermined system (see §20.1).

LAPACK also computes two nonstandard factorization of an $m \times n$ A:

$$\texttt{xGEQLF}: \ A = Q\begin{bmatrix} 0 \\ L \end{bmatrix}, \quad m \ge n, \qquad \texttt{xGERQF}: \ A = [0 \quad R]\,Q, \quad m \le n,$$

where L is lower trapezoidal and R upper trapezoidal.

Problems

18.1. Find the eigenvalues of a Householder matrix and a Givens matrix.

18.2. Let $\widehat{P} = I - \widehat{\beta}\widehat{v}\widehat{v}^T$, where $\widehat{\beta}$ and $\widehat{v}$ are the computed quantities described in Lemma 18.1. Derive a bound for $\|\widehat{P}^T\widehat{P} - I\|_2$.

18.3. A complex Householder matrix has the form

$$P = I - \beta vv^*,$$

where $0 \ne v \in \mathbb{C}^n$ and $b = 2/v^*v$. For given $x, y \in \mathbb{C}^n$, show how to determine, if possible, P so that $Px = y$.

18.4. (Wilkinson [1089, 1965, p. 242]) Let $x \in \mathbb{R}^n$ and let P be a Householder matrix such that $Px = \pm\|x\|_2 e_1$. Let $G_{1,2}, \ldots, G_{n-1,n}$ be Givens rotations such that $Qx := G_{1,2} \ldots G_{n-1,n}x = \pm\|x\|_2 e_1$. True or false: $P = Q$?

18.5. In the QR factorization with column pivoting, columns are interchanged at the start of the kth stage of the reduction so that, in the notation of (18.1), $\|x_k\|_2 \ge \|C_k(:, j)\|_2$ for all j. Show that the resulting R factor satisfies

$$r_{kk}^2 \ge \sum_{i=k}^{j} r_{ij}^2, \qquad j = k+1\colon n, \quad k = 1\colon n,$$

so that, in particular, $|r_{11}| \ge |r_{22}| \ge \ldots \ge |r_{nn}|$. (These are the same equations as (10.13), which hold for the Cholesky factorization with complete pivoting—why?)

18.6. Let $W \in \mathbb{R}^{m\times m}$ be a product of disjoint Givens rotations. Show that $\|\,|W|\,\|_2 \le \sqrt{2}$.

18.7. The CGS method and the MGS method applied to $A \in \mathbb{R}^{m\times n}$ ($m \ge n$) compute a QR factorization $A = QR$, $Q \in \mathbb{R}^{m\times n}$. Define the orthogonal projection $P_i = q_i q_i^T$, where $q_i = Q(:, i)$. Show that

$$(I - P_j)(I - P_{j-1}) \ldots (I - P_1) = I - P_j - \cdots - P_1.$$

Show that the CGS method corresponds to the operations

$$a_j \leftarrow (I - P_j - \cdots - P_1)a_j, \qquad j = 1\colon n,$$

while MGS corresponds to

$$a_j \leftarrow (I - P_j)(I - P_{j-1}) \dots (I - P_1)a_j, \qquad j = 1\colon n.$$

18.8. Let $A = [a_1, \ a_2] \in \mathbb{R}^{m \times 2}$ and denote the angle between a_1 and a_2 by q, $0 \le q \le \pi/2$. (Thus, $\cos\theta := |a_1^T a_2|/(\|a_1\|_2 \|a_2\|_2)$.) Show that

$$\kappa_2(A) \ge \frac{\max(\|a_1\|_2, \|a_2\|_2)}{\min(\|a_1\|_2, \|a_2\|_2)} \cot \theta.$$

18.9. (Björck [107, 1967]) Let

$$A = \begin{bmatrix} 1 & 1 & 1 \\ \epsilon & 0 & 0 \\ 0 & \epsilon & 0 \\ 0 & 0 & \epsilon \end{bmatrix},$$

which is a matrix of the form discussed by Lauchli [692, 1961]. Assuming that $fl(1 + \epsilon^2) = 1$, evaluate the Q matrices produced by the CGS and MGS methods and assess their orthonormality.

18.10. Show that the matrix P in (18.19) has the form

$$P = \begin{bmatrix} 0_n & Q^T \\ Q & I - QQ^T \end{bmatrix},$$

where Q is the matrix obtained from the MGS method applied to A.

18.11. (Björck and Paige [119, 1992]) For any matrices satisfying

$$\begin{bmatrix} \Delta A_1 \\ A + \Delta A_2 \end{bmatrix} = \begin{bmatrix} P_{11} \\ P_{21} \end{bmatrix} R, \qquad P_{11}^T P_{11} + P_{21}^T P_{21} = I,$$

where both P_{11} and P_{21} have at least as many rows as columns, show that there exists an orthonormal Q such that $A + DA = QR$, where

$$\Delta A = F \Delta A_1 + \Delta A_2, \qquad \|F\|_2 \le 1.$$

(Hint: use the CS decomposition $P_{11} = UCW^T$, $P_{21} = VSW^T$, where U and V have orthonormal columns, W is orthogonal, and C and S are square, nonnegative diagonal matrices with $C^2 + S^2 = I$. Let $Q = VW^T$. Note, incidentally, that $P_{21} = VW^T \cdot WSW^T$, so $Q = VW^T$ is the orthonormal polar factor of P_{21}, and hence is the nearest orthonormal matrix to P_{21} in the 2- and Frobenius norms. For details of the CS decomposition see Golub and Van Loan [470, 1989, pp. 77, 471] and Paige and Wei [814, 1994].)

18.12. We know that Householder QR factorization of $\begin{bmatrix} 0 \\ A \end{bmatrix}$ is equivalent to the MGS method applied to A, and Problem 18.10 shows that the orthonormal matrix Q from MGS is a submatrix of the orthogonal matrix P from the Householder method. Since Householder's method produces a nearly orthogonal P, does it not follow that MGS must also produce a nearly orthonormal Q?

18.13. (Higham [557, 1994]) Let $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ have the polar decomposition $A = UH$. Show that

$$\frac{\|A^TA - I\|_2}{\|A\|_2 + 1} \leq \|A - U\|_2 \leq \|A^TA - I\|_2.$$

This result shows that the two measures of orthonormality $\|A^TA - I\|_2$ and $\|A - U\|_2$ are essentially equivalent (cf. (18.22)).

Chapter 19

The Least Squares Problem

> *For some time it has been believed that orthogonalizing methods did not suffer this squaring of the condition number . . . It caused something of a shock, therefore, when in* 1966 *Golub and Wilkinson . . . asserted that already the multiplications QA and Qb may produce errors in the solution containing a factor* $x^2(A)$.
>
> — A. VAN DER SLUIS,
> *Stability of the Solutions of Linear Least Squares Problems* (1975)

> *Most packaged regression problems do compute a cross-products matrix and solve the normal equations using a matrix inversion subroutine. All the programs . . . that disagreed (and some of those that agreed) with the unperturbed solution tried to solve the normal equations.*
>
> — ALBERT E. BEATON, DONALD B. RUBIN, and JOHN L. BARONE,
> *The Acceptability of Regression Solutions: Another Look at Computational Accuracy* (1976)

> *On January* 1, 1801 *Giuseppe Piazzi discovered the asteroid Ceres. Ceres was only visible for forty days before it was lost to view behind the sun . . . Gauss, using three observations, extensive analysis, and the method of least squares, was able to determine the orbit with such accuracy that Ceres was easily found when it reappeared in late* 1801.
>
> — DAVID K. KAHANER, CLEVE B. MOLER, and STEPHEN G. NASH,
> *Numerical Methods and Software* (1989)

In this chapter we consider the least squares (LS) problem $\min_x \|b - Ax\|_2$, where $A \in \mathbb{R}^{m\times n}$ ($m > n$) has full rank. We begin by examining the sensitivity of the LS problem to perturbations. Then we examine the stability of methods for solving the LS problem, covering QR factorization methods, the normal equations and seminormal equations methods, and iterative refinement. Finally, we show how to compute the backward error of an approximate LS solution.

We do not develop the basic theory of the LS problem, which can be found in standard textbooks (see, for example, Golub and Van Loan [470, 1989, §5.3]). However, we recall the fundamental result that any solution of the LS problem satisfies the normal equations $A^TAx = A^Tb$ (see Problem 19.1). Therefore if A has full rank there is a unique LS solution. More generally, whatever the rank of A the vector $x_{LS} = A^+b$ is an LS solution, and it is the solution of minimal 2-norm. Here, A^+ is the pseudo-inverse of A (given by $A^+ = (A^TA)^{-1}A^T$ when A has full rank); see Problem 19.3. (For more on the pseudo-inverse see Stewart and Sun [954, 1990, §3.1].)

19.1. Perturbation Theory

Perturbation theory for the LS problem is, not surprisingly, more complicated than for linear systems, and there are several forms in which bounds can be stated. We begin with a normwise perturbation theorem that is a restatement of a result of Wedin [1069, 1973, Thm. 5.1]. For an arbitrary rectangular matrix A we define the condition number $\kappa_2(A) = \|A\|_2\|A^+\|_2$. If A has $r = \text{rank}(A)$ nonzero singular values, $\sigma_1 \ge \cdots \ge \sigma_r$, then $\kappa_2(A) = \sigma_1/\sigma_r$.

Theorem 19.1 (Wedin). *Let $A \in \mathbb{R}^{m\times n}$ ($m \ge n$) and $A + \Delta A$ both be of full rank, and let*

$$\begin{gathered} \|b - Ax\|_2 = \min, \qquad r = b - Ax, \\ \|(b+\Delta b) - (A+\Delta A)y\|_2 = \min, \qquad s = b + \Delta b - (A+\Delta A)y, \\ \|\Delta A\|_2 \le \epsilon\|A\|_2, \quad \|\Delta b\|_2 \le \epsilon\|b\|_2. \end{gathered}$$

Then, provided that $\kappa_2(A)\epsilon < 1$,

$$\frac{\|x-y\|_2}{\|x\|_2} \le \frac{\kappa_2(A)\epsilon}{1-\kappa_2(A)\epsilon}\left(2 + (\kappa_2(A)+1)\frac{\|r\|_2}{\|A\|_2\|x\|_2}\right) \tag{19.1}$$

$$\frac{\|r-s\|_2}{\|b\|_2} \le (1 + 2\kappa_2(A))\epsilon. \tag{19.2}$$

These bounds are approximately attainable. □

Proof. We defer a proof until § 19.8, since the techniques used in the proof are not needed elsewhere in this chapter. □

The bound (19.1) is usually interpreted as saying that the sensitivity of the LS problem is measured by $\kappa_2(A)$ when the residual is small or zero and by $\kappa_2(A)^2$ otherwise. This means that the sensitivity of the LS problem depends strongly on b as well as A, unlike for a square linear system.

Here is a simple example where the $\kappa_2(A)^2$ effect is seen:

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

It is a simple exercise to verify that

$$x = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ \frac{1}{2\epsilon} \end{bmatrix}, \quad s = \begin{bmatrix} 0 \\ -\frac{1}{2} \\ \frac{1}{2} \end{bmatrix}.$$

Since $\kappa_2(A) = 1/\epsilon$,

$$\begin{aligned} \frac{\|x-y\|_2}{\|x\|_2} &= \frac{1}{2\epsilon} \lesssim \kappa_2(A)^2 \frac{\|\Delta A\|_2}{\|A\|_2} = \frac{1}{\epsilon}, \\ \frac{\|r-s\|_2}{\|b\|_2} &= \frac{1}{2} \lesssim \kappa_2(A) \frac{\|\Delta A\|_2}{\|A\|_2} = 1. \end{aligned}$$

Surprisingly, it is easier to derive componentwise perturbation bounds than normwise ones for the LS problem. The key idea is to express the LS solution and its residual as the solution of the augmented system

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}, \tag{19.3}$$

which is simply another way of writing the normal equations, $A^TAx = A^Tb$. This is a square nonsingular system, so standard techniques can be applied. The perturbed system of interest is

$$\begin{bmatrix} I & A+\Delta A \\ (A+\Delta A)^T & 0 \end{bmatrix} \begin{bmatrix} s \\ y \end{bmatrix} = \begin{bmatrix} b+\Delta b \\ 0 \end{bmatrix}, \tag{19.4}$$

where we assume that

$$|\Delta A| \le \epsilon E, \quad |\Delta b| \le \epsilon f. \tag{19.5}$$

From (19.3) and (19.4) we obtain

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} s-r \\ y-x \end{bmatrix} = \begin{bmatrix} \Delta b - \Delta A y \\ -\Delta A^T s \end{bmatrix}.$$

Premultiplying by the inverse of the matrix on the left gives

$$\begin{bmatrix} s-r \\ y-x \end{bmatrix} = \begin{bmatrix} I-AA^+ & (A^+)^T \\ A^+ & -(A^TA)^{-1} \end{bmatrix} \begin{bmatrix} \Delta b - \Delta A y \\ -\Delta A^T s \end{bmatrix}. \tag{19.6}$$

Looking at the individual block components we obtain

$$|s-r| \le \epsilon\big(|I-A^+A|(f+E|y|) + |A^+|^T E^T |s|\big), \tag{19.7}$$

$$|y-x| \le \epsilon\big(|A^+|(f+E|y|) + |(A^TA)^{-1}|E^T|s|\big). \tag{19.8}$$

(Note that $\|I - AA^+\|_2 = \min\{1, m-n\}$, as is easily proved using the singular value decomposition (SVD).) On taking norms we obtain the desired perturbation result.

Theorem 19.2. *Let* $A \in \mathbb{R}^{m\times n}$ *(*$m \ge n$*) and* $A + \Delta A$ *be of full rank. For the perturbed LS problem described by (19.4) and (19.5) we have*

$$\frac{\|x-y\|}{\|x\|} \le \epsilon \frac{\|\,|A^+|(f+E|y|)\,\| + \|\,|(A^TA)^{-1}|E^T|s|\,\|}{\|x\|}, \tag{19.9}$$

$$\frac{\|r-s\|}{\|r\|} \le \epsilon \frac{\|\,|I-AA^+|(f+E|y|)\,\| + \|\,|A^+|^T E^T |s|\,\|}{\|r\|}, \tag{19.10}$$

for any monotonic norm. These bounds are approximately attainable. ☐

For a square system, we have $s = 0$, and we essentially recover Theorem 7.4. Note, however, that the bounds contain the perturbed vectors y and s. For theoretical analysis it may be preferable to use alternative bounds in which x and r replace y and s and there is an extra factor

$$\left(1 - \epsilon \left\| \begin{bmatrix} |I-AA^+| & |A^+|^T \\ |A^+| & |(A^TA)^{-1}| \end{bmatrix} \begin{bmatrix} 0 & E \\ E^T & 0 \end{bmatrix} \right\| \right)^{-1},$$

where the term in parentheses is assumed to be positive. For practical computation (19.9) is unsatisfactory because we do not know $s = b + \Delta b - (A + \Delta A)y$. However, as Stewart and Sun observe [954, 1990, p. 159], $\hat{r} = b - Ay$ is computable and

$$|s| \le |\hat{r}| + \epsilon(f + E|y|),$$

and using this bound in (19.9) makes only a second-order change.

The componentwise bounds enjoy better scaling properties than the normwise ones. If $E = |A|$ and $f = |b|$ then the bounds (19.7) and (19.8), and to a lesser extent (19.9) and (19.10), are invariant under column scalings $b - Ax \to b - AD \cdot D^{-1}x$ (D diagonal). Row scaling does affect the componentwise bounds, since it changes the LS solution, but the componentwise bounds are less sensitive to the row scaling than the normwise bounds, in a way that is difficult to make precise.

19.2. Solution by QR Factorization

Let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$ and rank(A) = n. If A has the QR factorization

$$Q^T A = \begin{bmatrix} R \\ 0 \end{bmatrix}$$

then

$$\|Ax - b\|_2^2 = \|Q^T Ax - Q^T b\|_2^2 =: \left\| \begin{bmatrix} Rx - c \\ -d \end{bmatrix} \right\|_2^2$$

It follows that the unique LS solution is $x = R^{-1}c$, and the residual $\|b - Ax\|_2 = \|d\|_2$. Thus the LS problem can be solved with relatively little extra work beyond the computation of a QR factorization. Note that Q is not required explicitly; we just need the ability to apply Q^t to a vector.

It is well known that the Givens and Householder QR factorization algorithms provide a normwise backward stable way to solve the LS problem. The next result expresses this fact for the Householder method and also provides componentwise backward error bounds (essentially the same result holds for the Givens method).

As in Chapter 18, we will use the generic constant γ_{cm}, in which c denotes a small integer. We will assume implicitly that a condition holds of the form $mn\gamma_{cm} < 1/2$.

Theorem 19.3. *Let $A \in \mathbb{R}^{m\times n}$ ($m \geq n$) have full rank and suppose the LS problem $\min_x \|b - Ax\|_2$ is solved using the Householder QR factorization method. The computed solution $\widehat{x}$ is the exact LS solution to*

$$\min_x \|(b + \Delta b) - (A + \Delta A)\widehat{x}\|_2,$$

where the perturbations satisfy the normwise bounds

$$\|\Delta A\|_F \leq n\gamma_{cm}\|A\|_F, \quad \|\Delta b\|_2 \leq n\gamma_{cm}\|b\|_2,$$

and the componentwise bounds

$$|\Delta A| \leq mn\gamma_{cm}G|A|, \qquad |\Delta b| \leq mn\gamma_{cm}G|b|,$$

where $\|G\|_F = 1$.

Proof. The proof is a straightforward generalization of the proof of Theorem 18.5 and is left as an exercise (Problem 19.2). ☐

As for Theorem 18.5 (see (18.7)), Theorem 19.3 remains true if we set $\Delta b \equiv 0$, but in general there is no advantage to restricting the perturbations to A.

Theorem 19.3 is a strong result, but it does not bound the residual of the computed solution, which, after all, is what we are trying to minimize. How close, then, is $\|b-A\widehat{x}\|_2$ to $\min_x \|b - Ax\|_2$? We can answer this question using the perturbation theory of §19.1. With $\widehat{r} := b + \Delta b - (A + \Delta A)\widehat{x}$, $x := x_{LS}$ and $r := b - Ax$, (19.6) yields

$$\widehat{r} - r = (I - AA^+)(\Delta b - \Delta A\widehat{x}) - (A^+)^T \Delta A^T \widehat{r},$$

so that

$$(b - A\widehat{x}) - r = -AA^+(\Delta b - \Delta A\widehat{x}) - (A^+)^T \Delta A^T \widehat{r}.$$

Substituting the bounds for ΔA and Δb from Theorem 19.3, and noting that $\|AA^+\|_2 = 1$, we obtain

$$\begin{aligned} \|(b - A\widehat{x}) - r\|_2 &\le mn\gamma_{cm}\big(\|G(|b| + |A||x|)\|_2 \\ &\qquad + \|\,|A^+|^T|A^T|G^T|r|\,\|_2\big) + O(u^2), \\ &\le mn\gamma_{cm}\big(\|\,|b| + |A||x|\,\|_2 + \mathrm{cond}_2(A^T)\|r\|_2\big) + O(u^2), \end{aligned}$$

where $\mathrm{cond}_2(A) := \|\,|A^+||A|\,\|_2$. Hence

$$\|b - A\widehat{x}\|_2 \le mn\gamma_{cm}\|\,|b| + |A||x|\,\|_2 + \big(1 + mn\gamma_{cm}\,\mathrm{cond}_2(A^T)\big)\|r\|_2 + O(u^2).$$

This bound contains two parts. The term $mn\gamma_{cm}\|\,|b| + |A||x|\,\|_2$ is a multiple of the bound for the error in evaluating $fl(b - Ax)$, and so is to be expected. The factor $1 + mn\gamma_{cm}\,\mathrm{cond}_2(A^T)$ will be less than 1.1 (say) provided that $\mathrm{cond}_2(A^T)$ is not too large. Note that $\mathrm{cond}_2(A^T) \le {}_{nk2}(A)$ and $\mathrm{cond}_2(A^T)$ is invariant under column scaling of A ($A \leftarrow A\,\mathrm{diag}(d_i)$, $d_i \ne 0$). The conclusion is that, unless A is very ill conditioned, the residual $b - A\widehat{x}$ will not exceed the larger of the true residual $r = b - Ax$ and a constant multiple of the error in evaluating $fl(r)$—a very satisfactory result.

19.3. Solution by the Modified Gram–Schmidt Method

The modified Gram-Schmidt (MGS) method can be used to solve the LS problem. However, we must not compute x from $x = R^{-1}(Q^Tb)$, because the lack of orthonormality of the computed $\widehat{Q}$ would adversely affect the stability. Instead we apply MGS to the augmented matrix $[A \;\; b]$:

$$[A \;\; b] = [Q_1 \;\; q_{n+1}]\begin{bmatrix} R & z \\ 0 & \rho \end{bmatrix}.$$

We have

$$Ax - b = [A \ \ b]\begin{bmatrix} x \\ -1 \end{bmatrix} = [Q_1 \ \ q_{n+1}]\begin{bmatrix} Rx - z \\ -\rho \end{bmatrix}$$
$$= Q_1(Rx - z) - \rho q_{n+1}.$$

Since q_{n+1} is orthogonal to the columns of Q_1, $\|b - Ax\|_2^2 = \|Rx - z\|_2^2 + \rho^2$, so the LS solution is $x = R^{-1}z$. Of course, $z = Q_1^T b$, but z is now computed as part of the MGS procedure instead of as a product between Q^T and b

Björck [107, 1967] shows that this algorithm is forward stable, in the sense that the forward error $\|x - \widehat{x}\|_2/\|x\|_2$ is as small as that for a normwise backward stable method. It has recently been shown by Björck and Paige [119, 1992] that the algorithm is, in fact, normwise backward stable (see also Björck [115, 1994]), that is, a normwise result of the form in Theorem 19.3 holds. Moreover, a componentwise result of the form in Theorem 19.3 holds too—see Problem 19.5. Hence the possible lack of orthonormality of $\widehat{Q}$ does not impair the stability of the MGS method as a means for solving the LS problem.

19.4. The Normal Equations

The oldest method of solving the LS problem is to form and solve the normal equations, $A^TAx = A^Tb$. Assuming that A has full rank, we can use the following procedure:

Form $C = A^TA$ and $c = A^Tb$.
Compute the Cholesky factorization $C = R^TR$.
Solve $R^Ty = c$, $Rx = y$.

Cost: : $n^2(m + n/3)$ flops.

If $m >> n$, the normal equations method requires about half as many flops as the Householder QR factorization approach (or the MGS method). However, it has less satisfactory numerical stability properties. There are two problems. The first is that information may be lost when $\widehat{C} := fl(A^TA)$ is formed-essentially because forming the cross product is a squaring operation that increases the dynamic range of the data. A simple example is the matrix

$$A = \begin{bmatrix} 1 & 1 \\ \epsilon & 0 \end{bmatrix}, \qquad 0 < \epsilon < \sqrt{u},$$

for which

$$A^TA = \begin{bmatrix} 1+\epsilon^2 & 1 \\ 1 & 1 \end{bmatrix}, \qquad fl(A^TA) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

Even though A is distance approximately ϵ from a rank-deficient matrix, and hence unambiguously full rank if $\epsilon \approx \sqrt{u/2}$, the computed cross product is singular. In general, whenever $\kappa_2(A) \geq u^{-1/2}$ we can expect $\widehat{C}$ to be singular or indefinite, in which case Cholesky factorization is likely to break down (Theorem 10.7).

The second weakness of the normal equations method is more subtle and is explained by a rounding error analysis. In place of $C = A^TA$ and $c = A^Tb$ we compute

$$\begin{aligned} \widehat{C} &= A^TA + \Delta C_1, \qquad |\Delta C_1| \leq \gamma_m |A^T||A|, \\ \widehat{c} &= A^Tb + \Delta c, \qquad |\Delta c| \leq \gamma_m |A^T||b|. \end{aligned}$$

By Theorems 10.3 and 10.4, the computed Cholesky factor $\widehat{R}$ and solution $\widehat{x}$ satisfy

$$\begin{aligned} \widehat{R}^T\widehat{R} &= \widehat{C} + \Delta C_2, \qquad |\Delta C_2| \leq \gamma_{n+1} |\widehat{R}^T||\widehat{R}|, \\ (\widehat{C} + \Delta C_3)\widehat{x} &= \widehat{c}, \qquad |\Delta C_3| \leq 2\gamma_{n+1} |\widehat{R}^T||\widehat{R}|. \end{aligned} \tag{19.11}$$

Overall,

$$\begin{gathered} (A^TA + \Delta A)\widehat{x} = A^Tb + \Delta c, \\ |\Delta A| \leq \gamma_m |A^T||A| + 2\gamma_{n+1} |\widehat{R}^T||\widehat{R}|, \quad |\Delta c| \leq \gamma_m |A^T||b|. \end{gathered} \tag{19.12}$$

By bounding $\| \, |\widehat{R}^T||\widehat{R}| \, \|_2$ with the aid of (19.11), we find that

$$\|\Delta A\|_2 \leq (mn + 2n^2 + 2n)u\|A\|_2^2 + O(u^2), \tag{19.13a}$$

$$\|\Delta c\|_2 \leq mn^{1/2}u\|A\|_2\|b\|_2 + O(u^2). \tag{19.13b}$$

These bounds show that we have solved the normal equations in a backward stable way, as long as $\|A\|_2\|b\|_2 \approx \|A^Tb\|_2$. But if we try to translate this result into a backward error result for the LS problem itself, we find that the best backward error bound contains a factor $\kappa_2(A)$ [569, 1987]. The best forward error bound we can expect, in view of (19.13), is of the form

$$\frac{\|x - \widehat{x}\|_2}{\|x\|_2} \lesssim c_{m,n}\kappa_2(A)^2 u \tag{19.14}$$

(since $\kappa_2(A^TA) = \kappa_2(A)^2$). Now we know from Theorem 19.1 that the sensitivity of the LS problem is measured by $\kappa_2(A)^2$ if the residual is large, but by $\kappa_2(A)$ if the residual is small. It follows that the normal equations method has a forward error bound that can be much larger than that possessed by a backward stable method.

A mitigating factor for the normal equations method is that, in view of Theorem 10.6, we can replace (19.14) by the (not entirely rigorous) bound

$$\frac{\|D(x - \widehat{x})\|_2}{\|Dx\|_2} \lesssim c_{m,n}\kappa_2(B)^2 u,$$

where $A = BD$, with $D = \mathrm{diag}(\|A(:, i)\|_2)$, so that B has columns of unit 2-norm. Van der Sluis's result (Theorem 7.5) shows that

$$\kappa_2(B) \le \sqrt{n} \min_{F \text{ diagonal}} \kappa_2(AF).$$

Hence the normal equations method is, to some extent, insensitive to poor column scaling of A.

Although numerical analysts almost invariably solve the full rank LS problem by QR factorization, statisticians frequently use the normal equations (though perhaps less frequently than they used to, thanks to the influence of numerical analysts). The normal equations do have a useful role to play. In many statistical problems the regression matrix is contaminated by errors of measurement that are very large relative to the roundoff level; the effects of rounding errors are then likely to be insignificant compared with the effects of the measurement errors, especially if IEEE double precision (as opposed to single precision) arithmetic is used.

The normal equations (NE) versus (Householder) QR factorization debate can be summed up as follows.

- The two methods have a similar computational cost if $m \approx n$, but the NE method is up to twice as fast for $m \gg n$. (This statement assumes that A and b are dense; for details of the storage requirements and computational cost of each method for sparse matrices, see, for example, Björck [116, 1996] and Heath [510, 1984].)
- The QR method is always backward stable. The NE method is guaranteed to be backward stable only if A is well conditioned.
- The forward error from the NE method can be expected to exceed that for the QR method when A is ill conditioned and the residual of the LS problem is small.
- The QR method lends itself to iterative refinement, as described in the next section. Iterative refinement can be applied to the NE method, but the rate of convergence inevitably depends on $\kappa_2(A)^2$ instead of $\kappa_2(A)$.

19.5. Iterative Refinement

As for square linear systems, iterative refinement can be used to improve the accuracy and stability of an approximate LS solution. However, for the LS problem there is more than one way to construct an iterative refinement scheme.

By direct analogy with the square system case, we might consider the scheme

1. Compute $r = b - A\widehat{x}$.
2. Solve the LS problem $\min_d \|Ad - r\|_2$.
3. Update $y = \widehat{x} + d$.

(Repeat from step 1 if necessary, with $\widehat{x}$ replaced by y).

This scheme is certainly efficient-a computed QR factorization (for example) of A can be reused at each step 2. Golub and Wilkinson [466, 1966] investigated this scheme and found that it works well only for nearly consistent systems.

An alternative approach suggested by Björck [106, 1967] is to apply iterative refinement to the augmented system (19.3), so that both x and r are refined simultaneously. Since this is a square, nonsingular system, existing results on the convergence and stability of iterative refinement can be applied, and we would expect the scheme to work well. To make precise statements we need to examine the augmented system method in detail.

For the refinement steps we need to consider an augmented system with an arbitrary right-hand side:

$$r + Ax = f, \tag{19.15a}$$

$$A^T r = g. \tag{19.15b}$$

If A has the QR factorization

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix},$$

where $R \in \mathbb{R}^{n\times n}$, then (19. 15) transforms to

$$Q^T r + \begin{bmatrix} R \\ 0 \end{bmatrix} x = Q^T f,$$

$$[\,R^T \quad 0\,]\, Q^T r = g.$$

This system can be solved as follows:

$$h = R^{-T} g,$$

$$d = Q^T f = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix},$$

$$r = Q\begin{bmatrix} h \\ d_2 \end{bmatrix},$$

$$x = R^{-1}(d_1 - h).$$

The next result describes the effect of rounding errors on the solution process.

Theorem 19.4. *Let* $A \in \mathbb{R}^{m\times n}$ *be of full rank* $n \le m$ *and suppose the augmented system* (19.15) *is solved using a Householder QR factorization of A as described above. The computed* $\widehat{x}$ *and* $\widehat{r}$ *satisfy*

$$\begin{bmatrix} I & A+\Delta A_1 \\ (A+\Delta A_2)^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{r} \\ \widehat{x} \end{bmatrix} = \begin{bmatrix} f+\Delta f \\ g+\Delta g \end{bmatrix},$$

where

$$\begin{aligned} |\Delta A_i| &\le mn\gamma_{cm} G|A|, \qquad i = 1\colon 2, \\ |\Delta f| &\le \sqrt{m}n\gamma_{cm}(H_1|f| + H_2|\widehat{r}|), \\ |\Delta g| &\le \sqrt{m}n\gamma_{cm}|A^T|H_3|\widehat{r}|, \end{aligned}$$

with $\|G\|_F = 1$, $\|H_i\|_F = 1$, $i = 1{:}3$.

Proof. The proof does not involve any new ideas and is rather tedious, so we omit it. $\square$

Consider first fixed precision iterative refinement. Theorem 19.4 implies that the computed solution $(\widehat{r}^T, \widehat{x}^T)^T$ to (19.15) satisfies

$$\left| \begin{bmatrix} f \\ g \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{r} \\ \widehat{x} \end{bmatrix} \right| \le mn\gamma_{cm} \left(\begin{bmatrix} H_2 & G|A| \\ |A^T|(G^T + H_3) & 0 \end{bmatrix} \begin{bmatrix} |\widehat{r}| \\ |\widehat{x}| \end{bmatrix} + \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} |f| \\ |g| \end{bmatrix} \right).$$

Unfortunately, this bound is not of a form that allows us to invoke Theorem 11.4. However, we can apply Theorem 11.3, which tells us that the corrected solution $(\widehat{s}^T, \widehat{y}^T)^T$ obtained after one step of iterative refinement satisfies

$$\begin{aligned} \left| \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{s} \\ \widehat{y} \end{bmatrix} \right| \le{} & mn\gamma_{cm} \begin{bmatrix} H_1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} |\widehat{r}_1| \\ |\widehat{r}_2| \end{bmatrix} + u \begin{bmatrix} I & |A| \\ |A^T| & 0 \end{bmatrix} \begin{bmatrix} |\widehat{s}| \\ |\widehat{y}| \end{bmatrix} \\ & + \gamma_{m+n+1} \left(\begin{bmatrix} I & |A| \\ |A^T| & 0 \end{bmatrix} \begin{bmatrix} |\widehat{s}| \\ |\widehat{y}| \end{bmatrix} + \begin{bmatrix} |b| \\ 0 \end{bmatrix} \right) \\ & + O(u^2). \end{aligned} \tag{19.16}$$

Here, $(\widehat{r}_1^T, \widehat{r}_2^T)^T$ denotes the residual of the augmented system corresponding to the original computed solution. We will make two simplifications to the bound (19.16). First, since $(\widehat{r}_1^T, \widehat{r}_2^T)^T = O(u)$, the first term in the bound may be included in the $O(u^2)$ term. Second, (19.16) yields $|b - \widehat{s} - A\widehat{y}| = O(u)$ and so $|\widehat{s}| \le |A||\widehat{y}| + |b| + O(u)$. With these two simplifications, (19.16) may be written

$$\left| \begin{bmatrix} b \\ 0 \end{bmatrix} - \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{s} \\ \widehat{y} \end{bmatrix} \right| \le 2\gamma_{m+n+2} \left(\begin{bmatrix} 0 & |A| \\ |A^T| & 0 \end{bmatrix} \begin{bmatrix} |\widehat{s}| \\ |\widehat{y}| \end{bmatrix} + \begin{bmatrix} |b| \\ 0 \end{bmatrix} \right) + O(u^2).$$

In view of the Oettli–Prager result (Theorem 7.3) this inequality tells us that, asymptotically, the solution $\widehat{y}$ produced after one step of fixed precision iterative refinement has a small componentwise relative backward error with respect to the augmented system. However, this backward error allows the two occurrences of A in the augmented system coefficient matrix to be perturbed differently, and thus is not a true componentwise backward error for the LS problem. Nevertheless, the result tells us that iterative refinement can be expected to produce some improvement in stability. Note that the bound in Theorem 19.2 continues to hold if we perturb the two occurrences of A in the augmented system differently. Therefore the bound is applicable to iterative refinement (with $E = |A|$, $f = |b|$), and so we can expect iterative refinement to mitigate the effects of poor row and column scaling of A. Numerical experiments show that these predictions are borne out in practice [549, 1991].

Turning to mixed precision iterative refinement, we would like to apply the analysis of \$11.1, with "$Ax = b$" again identified with the augmented system. However, the analysis of §11.1 requires a backward error result in which only the coefficient matrix is perturbed (see (11.1)). This causes no difficulties because from Theorem 19.4 we can deduce a normwise result (cf. Problem 7.7):

$$\left(\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} + \Delta\right)\begin{bmatrix} \widehat{r} \\ \widehat{x} \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \qquad \|\Delta\|_2 \le c_{m,n}u\left\|\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}\right\|_2.$$

The theory of §11.1 then tells us that mixed precision iterative refinement will converge as long as the condition number of the augmented system matrix is not too large and that the rate of convergence depends on this condition number. How does the condition number of the augmented system relate to that of A? Consider the matrix that results from the scaling $A \leftarrow \alpha^{-1}A$ ($\alpha > 0$):

$$C(\alpha) = \begin{bmatrix} \alpha I & A \\ A^T & 0 \end{bmatrix}. \tag{19.17}$$

Björck [106, 1967] shows that the eigenvalues of $C(\alpha)$ are

$$\lambda(C(\alpha)) = \begin{cases} \dfrac{\alpha}{2} \pm \left(\dfrac{\alpha^2}{4} + \sigma_i^2\right)^{1/2}, & i = 1\colon n, \\ 0, & m - n \text{ times}, \end{cases} \tag{19.18}$$

where σ_i, $i = 1\colon n$, are the singular values of A, and that

$$\sqrt{2}\kappa_2(A) \le \min_\alpha \kappa_2(C(\alpha)) \le 2\kappa_2(A), \quad \max_\alpha \kappa_2(C(\alpha)) > \kappa_2(A)^2, \tag{19.19}$$

with $\min_\alpha \kappa_2(C(\alpha))$ being achieved for $\alpha = \sigma_n/\sqrt{2}$ (see Problem 19.7). Hence $C(\alpha)$ may be much more ill conditioned than A. However, in our analysis we are at liberty to take $\min_\alpha \kappa_2(C(\alpha))$ as the condition number, because scaling the LS problem according to $b - Ax \leftarrow (b - Ax)/\alpha$ does not change the computed solution or the rounding errors in any way (at least not if α is a power of the machine base). Therefore it is, $\kappa_2(A)$ that governs the behaviour of mixed precision iterative refinement, irrespective of the size of the LS residual. As Björck [111, 1990] explains, this means that "in a sense iterative refinement is *even more satisfactory* for large residual least-squares problems." He goes on to explain that "When residuals to the augmented system are accumulated in precision β^{-t_2}, $t_2 \geq 2t_1$, this scheme gives solutions to full single-precision accuracy even though the initial solution may have no *correct significant figures.*"

Iterative refinement can be applied with the MGS method. Björck [106, 1967] gives the details and shows that mixed precision refinement works just as well as it does for Householder's method.

19.6. The Seminormal Equations

When we use a QR factorization to solve an LS problem $\min_x \|b - Ax\|_2$, the solution x is determined from the equation $Rx = Q^Tb$ (or via a similar expression involving Q for the MGS method). But if we need to solve for several right-hand sides that are not all available when the QR factorization is computed, we need to store Q before applying it to the right-hand sides. If A is large and sparse it is undesirable to store Q, as it can be much more dense than A. We can, however, rewrite the normal equations as

$$R^TRx = A^Tb,$$

which are called the *seminormal equations.* The solution x can be determined from these equations without the use of Q. Since the cross product matrix A^TA is not formed explicitly and R is determined stably via a QR factorization, we might expect this approach to be more stable than the normal equations method.

Björck [110, 1987] has done a detailed error analysis of the seminormal equations (SNE) method, under the assumption that R is computed by a backward stable method. His forward error bound for the SNE method is of the same form as that for the normal equations method, involving a factor $\kappa_2(A)^2$. Thus the SNE method is not backward stable. Björck considers applying one step of fixed precision iterative refinement to the SNE method, and he calls the resulting process the corrected seminormal equations (CSNE) method:

$$\begin{aligned} &R^TRx = A^Tb \\ &r = b - Ax \end{aligned}$$

$$R^TRw = A^Tr$$
$$y = x + w$$

It is important that the normal equations residual be computed as shown, as $A^T(b - Ax)$, and not as $A^Tb - A^TAx$. Björck derives a forward error bound for the CSNE method that is roughly of the form

$$\frac{\|x-y\|_2}{\|x\|_2} \le c_{m,n}\left(\kappa_2(A)u \cdot \kappa_2(A)^2u\left(1+\frac{\|b\|_2}{\|A\|_2\|x\|_2}\right)+\frac{\kappa_2(A)^2u\|r\|_2}{\|A\|_2\|x\|_2}\right).$$

Hence, if $\kappa_2(A)^2u \le 1$, the CSNE method has a forward error bound similar to that for a backward stable method, and the bound is actually smaller than that for the QR method if $\kappa_2(A)^2u \ll 1$ and r is small. However, the CSNE method is not backward stable for all A.

19.7. Backward Error

Although it has been known since the 1960s that a particular method for solving the LS problem, namely the Householder QR factorization method, yields a small normwise backward error (see §19.2), it was for a long time an open problem to obtain a formula for the backward error of an arbitrary approximate solution. Little progress had been made towards solving this problem until Waldén, Karlson, and Sun [1060, 1995] found an extremely elegant solution. We will denote by $\lambda_{\min}$ and $\sigma_{\min}$ the smallest eigenvalue of a symmetric matrix and the smallest singular value of a general matrix, respectively.

Theorem 19.5 (Waldén, Karlson, and Sun). *Let* $A \in \mathbb{R}^{m\times n}$ *(*$m \ge n$*),* $b \in \mathbb{R}^m$*,* $0 \ne y \in \mathbb{R}^n$*, and* $r = b - Ay$*. The normwise backward error*

$$\eta_F(y) := \min\{\, \|[\Delta A,\ \theta\Delta b]\|_F : \|(A+\Delta A)y - (b+\Delta b)\|_2 = \min \,\} \tag{19.20}$$

is given by

$$\eta_F(y) = \begin{cases} \dfrac{\|r\|_2}{\|y\|_2}\sqrt{\mu}, & \lambda_* \ge 0, \\ \left(\dfrac{\|r\|_2^2}{\|y\|_2^2}\mu + \lambda_*\right)^{1/2}, & \lambda_* < 0, \end{cases}$$

where

$$\lambda_* = \lambda_{\min}\left(AA^T - \mu\frac{rr^T}{\|y\|_2^2}\right), \quad \mu = \frac{\theta^2\|y\|_2^2}{1+\theta^2\|y\|_2^2}.$$

The backward error (19.20) is not a direct generalization of the usual normwise backward error for square linear systems, because it minimizes

Table 19.1. *LS backward errors and residual for Vandermonde system.*

	$\frac{\eta_F}{\|[A,b]\|_F}$ $(\theta = 1)$	$\frac{\eta_F}{\|A\|_F}$ $(\theta = \infty)$	$\|r\|_2$
NE	8.6×10^{-12}	2.40×10^{-10}	6.3×10^{-8}
QR	8.6×10^{-18}	2.41×10^{-16}	6.2×10^{-14}
MGS	2.6×10^{-18}	7.32×10^{-17}	2.0×10^{-14}
MGS(bad)	2.0×10^{-11}	5.66×10^{-10}	5.8×10^{-7}

$\|[\Delta A,\ \theta\Delta b]\|_F$ instead of $\max\{\|\Delta A\|_2/\|E\|_2, \|\Delta b\|_2/\|f\|_2\}$. However, the Parameter θ allows us some flexibility: taking the limit $\theta \to \infty$ forces $\Delta b = 0$, giving the case where only A is perturbed.

Theorem 19.5 can be interpreted as saying that if $\lambda_* \geq 0$ then the backward error is essentially that given by Theorem 7.1 for a consistent system. If $\lambda_* < 0$, however, the nearest perturbed system of which y is the LS solution is inconsistent. A sufficient condition for $\lambda_* < 0$ is $b \notin \operatorname{range}(A)$ (assuming $\mu \neq 0$), that is, the original system is inconsistent.

The formulae given in Theorem 19.5 are unsuitable for computation because they can suffer from catastrophic cancellation when $\lambda_* < 0$. Instead, the following alternative formula derived in [1060, 1995] should be used (see Problem 19.9):

$$\eta_F(y) = \min\{\,\eta_1,\ \sigma_{\min}([A\ \ \eta_1 C])\,\}, \tag{19.21a}$$

$$\eta_1 = \frac{\|r\|_2}{\|y\|_2}\sqrt{\mu}, \qquad C = I - \frac{rr^T}{r^T r}. \tag{19.21b}$$

To illustrate Theorem 19.5, we consider an LS problem with a 25 x 15 Vandermonde matrix $A = (p_j^{i-1})$, where the p_i are equally spaced on [0, 1], and $b = Ax$ with $x_i = i$ (giving a zero residual problem). The condition number $\kappa_2(A) = 1.47 \times 10^9$. We solved the LS problem in MATLAB in four different ways: by using the NE with Cholesky factorization, via Householder QR factorization, and via the MGS method, using both the stable approach described in §19.3 and the unstable approach in which Q^Tb is formed as a matrix–vector product (denoted MGS(bad)). The results, including the norms of the residuals $r = b - A\widehat{x}$, are shown in Table 19.1. As would be expected from the analysis in this chapter, the QR and stable MGS methods produce backward stable solutions, but the NE method and the unstable MGS approach do not.

As we saw in the analysis of iterative refinement, sometimes we need to consider the augmented system with different perturbations to A and AT. The next result shows that from the point of view of normwise perturbations and componentwise perturbations bounded by $|\Delta A| \le \varepsilon G|A|$, the lack of "symmetry" in the perturbations has little effect on the backward error of y.

Lemma 19.6 (Kiełbasiński and Schwetlick). *Let $A \in \mathbb{R}^{m\times n}$ $(m \ge n)$ and consider the perturbed augmented system*

$$\begin{bmatrix} I & A+\Delta A_1 \\ (A+\Delta A_2)^T & 0 \end{bmatrix} \begin{bmatrix} s \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}.$$

There is a vector $\widehat{s}$ and a perturbation DA with

$$DA = G_1 DA_1 + G_2 DA_2, \qquad G_1^T = G_1 = G_1^2, \quad G_1 + G_2 = I, \tag{19.22}$$

such that

$$\begin{bmatrix} I & A+\Delta A \\ (A+\Delta A)^T & 0 \end{bmatrix} \begin{bmatrix} \widehat{s} \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix},$$

that is, y solves the LS problem $\min_x \|(A + DA)x - b\|_2$.

Proof. If $s = b - (A + DA_1)y = 0$ we take $DA = DA_1$. Otherwise, we set

$$DA := PDA_2 + (I - P)DA_1 =: DA_1 + PH,$$

where $P = ss^T/s^Ts$ and $H = DA_2 - DA_1$. We have

$$\widehat{s} := b - (A+\Delta A)y = b - (A+\Delta A_1)y - PHy = \beta s,$$

where $\beta = 1 - s^T Hy/s^T s$. Then

$$\begin{aligned} (A+\Delta A)^T\widehat{s} &= (A+\Delta A_2 + (P-I)H)^T \beta s \\ &= \beta\big((A+\Delta A_2)^T s + H^T(P-I)s\big) = 0. \qquad \square \end{aligned}$$

Note that (19.22) implies a bound stronger than just $\|DA\|_2 \le \|DA_1\|_2 + \|DA_2\|_2$:

$$\|\Delta A\|_p \le (\|\Delta A_1\|_p^2 + \|\Delta A_2\|_p^2)^{1/2}, \qquad p = 2, F.$$

Turning to componentwise backward error, the simplest approach is to apply the componentwise backward error $\omega_{E,f}(y)$ to the augmented system (19.3), setting

$$E = \begin{bmatrix} 0 & E_A \\ E_A^T & 0 \end{bmatrix}$$

so as not to perturb the diagonal blocks I and O of the augmented system coefficient matrix. However, this approach allows A and AT to undergo different perturbations DA_1 and DA_2 with $\Delta A_1 \ne \Delta A_2^T$ and thus does not give a true backward error, and Lemma 19.6 is of no help. This problem can be overcome by using a structured componentwise backward error to force symmetry of the perturbations; see Higham and Higham [527, 1992] for details. One problem remains: as far as the backward error of y is concerned, the vector r in the augmented system is a vector of free parameters, so to obtain the true componentwise backward error we have to minimize the structure-preserving componentwise backward error over all r. This is a nonlinear optimization problem to which no closed-form solution is known. Experiments show that when y is a computed LS solution, $r = b - Ay$ is often a good approximation to the minimizing r [527, 1992], [549, 1991].

19.8. Proof of Wedin's Theorem

In this section we give a proof of Theorem 19.1. We define $P_A := AA^+$, the orthogonal projection onto range(A).

Lemma 19.7. *Let $A, B \in \mathbb{R}^{m\times n}$. If rank(A) = rank(B) and $h = \|A^+\|_2\|A-B\|_2 < 1$, then*

$$\|B^+\|_2 \le \frac{1}{1-\eta}\|A^+\|_2.$$

Proof. Let r = rank(A). A standard result on the perturbation of singular values gives

$$\sigma_r(B) \ge \sigma_r(A) - \|A - B\|_2,$$

that is,

$$\|B^+\|_2^{-1} \ge \|A^+\|_2^{-1} - \|A - B\|_2 > 0,$$

which gives the result on rearranging. □

Lemma 19.8. *Let $A, B \in \mathbb{R}^{m\times n}$. If* rank($A$) = rank($B$) *then*

$$\|P_A(I - P_B)\|_2 = \|P_B(I - P_A)\|_2 \le \|A - B\|_2 \min\{\, \|A^+\|_2, \|B^+\|_2 \,\}.$$

Proof. We have

$$\begin{aligned}\|P_B(I - P_A)\|_2 &= \|(P_B(I - P_A))^T\|_2\\ &= \|(I - P_A)P_B\|_2\\ &= \|(I - AA^+)BB^+\|_2\\ &= \|(I - AA^+)(A + (B - A))B^+\|_2\\ &= \|(I - AA^+)(B - A)B^+\|_2\\ &\le \|A - B\|_2\|B^+\|_2.\end{aligned}$$

The result then follows from the (nontrivial) equality $\|P_A(I - P_B)\|_2 = \|P_B(I-P_A)\|_2$; see, for example, Stewart [943, 1977, Thin. 2.3] or Stewart and Sun

[954, 1990, Lem. 3.3.5], where proofs that use the CS decomposition are given. $\square$

Proof of Theorem 19.1. Let $B := A + AA$. We have, with $r = b - Ax$,

$$\begin{aligned} y - z &= B^+(b + Db) - z = B^+(r + Ax + Db) - x \\ &= B^+(r + Bx - DAx + Db) - x \\ &= B^+(r - DAx + Db) - (I - B^+B)x \\ &= B^+(r - DAx + Db), \end{aligned} \tag{19.23}$$

since B has full rank. Now

$$B^+r = B^+(BB^+)r = B^+P_Br = B^+P_B(I - P_A)r. \tag{19.24}$$

Applying Lemmas 19.7 and 19.8, we obtain

$$\begin{aligned} \|B^+r\|_2 &\le \|B^+\|_2(\|B - A\|_2\|A^+\|_2)\|r\|_2 \\ &\le \frac{\|A^+\|_2}{1 - \|A^+\|_2\|\Delta A\|_2}\|\Delta A\|_2\|A^+\|_2\|r\|_2 \\ &= \frac{\kappa_2(A)^2\epsilon}{1 - \kappa_2(A)\epsilon}\frac{\|r\|_2}{\|A\|_2}. \end{aligned} \tag{19.25}$$

Similarly,

$$\begin{aligned} \|B^+(-\Delta Ax + \Delta b)\|_2 &\le \frac{\|A^+\|_2}{1 - \kappa_2(A)\epsilon}\epsilon(\|A\|_2\|x\|_2 + \|b\|_2) \\ &= \frac{\kappa_2(A)\epsilon}{1 - \kappa_2(A)\epsilon}\left(1 + \frac{\|b\|_2}{\|A\|_2\|x\|_2}\right)\|x\|_2 \\ &\le \frac{\kappa_2(A)\epsilon}{1 - \kappa_2(A)\epsilon}\left(2 + \frac{\|r\|_2}{\|A\|_2\|x\|_2}\right)\|x\|_2. \end{aligned} \tag{19.26}$$

The bound for $\|x - y\|_2/\|x\|_2$ is obtained by using inequalities (19.25) and (19.26) in (19.23).

Turning to the residual, using (19.23) we find that

$$\begin{aligned} s - r &= Db + B(x - y) - DAx \\ &= Db - BB^+(r - DAx + Db) - DAx \\ &= (I - BB^+)(Db - DAx) - BB^+r. \end{aligned}$$

Since $\|I - BB^+\|_2 = \min\{1, m - n\}$,

$$\|r - s\|_2 \le \|\Delta b\|_2 + \|\Delta Ax\|_2 + \|BB^+r\|_2 \le \epsilon(\|b\|_2 + \|A\|_2\|x\|_2) + \|BB^+r\|_2.$$

Using (19.24), Lemma 19.8, and $\|BB^+\|_2 = 1$, we obtain

$$\|BB^+r\|_2 = \|BB^+P_B(I - P_A)r\|_2 \le \|\Delta A\|_2\|A^+\|_2\|r\|_2 \le \kappa_2(A)\epsilon\|r\|_2.$$

Hence

$$\frac{\|r-s\|_2}{\|b\|_2} \le \epsilon\left(1+\frac{\|A\|_2\|x\|_2}{\|b\|_2}+\kappa_2(A)\frac{\|r\|_2}{\|b\|_2}\right) \le \epsilon(1+2\kappa_2(A)).$$

For the attainability, see Wedin [1069, 1973, §6]. □

Note that, as the proof has shown, Wedin's theorem actually holds without any restriction on m and n, provided we define $x = A^+b$ and $y = (A+ \Delta A)^+(b + \Delta b)$ when $m < n$ (in which case $r = 0$). We consider underdetermined systems in detail in the next chapter. The original version of Wedin's theorem also requires only rank(A) = rank($A + \Delta A$) and not that A have full rank.

19.9. Notes and References

The most comprehensive and up to date treatment of the LS problem is the book by Björck [116, 1996], which is an updated and expanded version of [112, 1990]. It treats many aspects not considered here, including rank-deficient, weighted, and constrained problems. An early book devoted to numerical aspects of the LS problem was written by Lawson and Hanson [695, 1974], who, together with Stewart [94 1, 1973], were the first to present error analysis for the LS problem in textbook form.

The history of the LS problem is described in the statistically oriented book by Farebrother [363, 1988].

The pseudo-inverse A^+ underlies the theory of the LS problem, since the LS solution can be expressed as $x = A^+b$. An excellent reference for perturbation theory of the pseudo-inverse is Stewart and Sun [954, 1990, §3.3]. The literature on pseudo-inverses is vast, as evidenced by the annotated bibliography of Nashed and Rail [786, 1976], which contains 1,776 references published up to 1976.

Normwise perturbation theory for the LS problem was developed by various authors in the 1960s and 1970s. The earliest analysis was by Golub and Wilkinson [466, 1966], who gave a first-order bound and were the first to recognize the potential $\kappa_2(A)^2$ sensitivity. A nonasymptotic perturbation bound was given by Björck [107, 1967], who worked from the augmented system.

An early set of numerical experiments on the Householder, Gram-Schmidt, and normal equations methods for solving the LS problem was presented by Jordan [618, 1968]; this paper illustrates the incomplete understanding of perturbation theory and error analysis for the LS problem at that time.

van der Sluis [1041, 1975] presents a geometric approach to LS perturbation theory and gives lower bounds for the effect of worst-case perturbations. Golub and Van Loan [470, 1989, Thin. 5.3.1] give a first-order analogue of Theorem 19.1 expressed in terms of the angle θ between b and range(A) instead of the residual r.

Wei [1072, 1990] gives a normwise perturbation result for the LS problem with a rank deficient A that allows $\mathrm{rank}(A + \Delta A) > \mathrm{rank}(A)$.

Componentwise perturbation bounds of the form in Theorem 19.2 were first derived by Björck in 1988 and variations have been given by Arioli, Duff, and de Rijk [25, 1989], Björck [113, 1991], and Higham [542, 1990].

Higham [542, 1990] examined the famous test problem from Longley [711, 1967]—a regression problem which has a notoriously ill-conditioned 16 x 7 coefficient matrix with $\kappa_2(A) \approx 5 \times 10^9$. The inequality (19.8) was found to give tight bounds for the effect of random componentwise relative perturbations of the problem generated in experiments of Beaton, Rubin, and Barone [86, 1976]. Thus componentwise perturbation bounds are potentially useful in regression analysis as an alternative to the existing statistically based techniques.

The tools required for a direct proof of the normwise backward error result in Theorem 19.3 are developed in Wilkinson's book *The Algebraic Eigenvalue Problem* [1089, 1965]. Results of this form were derived informally by Golub and Wilkinson (assuming the use of extended precision inner products) [466, 1966], stated by Wilkinson [1090, 1965, p. 93] and Stewart [941, 1973], and proved by Lawson and Hanson [695, 1974, Chap. 16].

The idea of using QR factorization to solve the LS problem was mentioned in passing by Householder [586, 1958]. Golub [463, 1965] worked out the details, using Householder QR factorization, and this method is sometimes called "Golub's method". In the same paper, Golub suggested the form of iterative refinement described at the start of §19.5 (which is implemented in a procedure by Businger and Golub [167, 1965]), and showed how to use QR factorization to solve an LS problem with a linear constraint $Bx = c$.

It was Björck [106, 1967] who first recognized that iterative refinement should be applied to the augmented system for best results, and he gave a detailed rounding error analysis for the use of a QR factorization computed by the Householder or MGS methods. Björck and Golub [118, 1967] give an Algol code for computation and refinement of the solution to an LS problem with a linear constraint; they use Householder transformations, while Björck [108, 1968] gives a similar code baaed on the Gram–Schmidt method. In [109, 1978], Björck dispels some misconceptions of statisticians about (mixed precision) iterative refinement for the LS problem; he discusses standard refinement together with two versions of refinement based on the seminormal equations.

Error analysis for solution of the LS problem by the classical Gram-Schmidt method with reorthogonalization is given by Abdelmalek [2, 1971], who obtains a forward error bound as good as that for a backward stable method.

Higham and Stewart [569, 1987] compare the normal equations method with the QR factorization method, with emphasis on aspects relevant to regression problems in statistics.

Foster [398, 1991] proposes a class of methods for solving the LS problem that are intermediate between the normal equations method and the MGS method, and that can be viewed as block MGS algorithms.

The most general analysis of QR factorization methods for solving the LS and related problems is by Björck and Paige [120, 1994], who consider an augmented system with an arbitrary right-hand side (see Problem 20.1) and prove a number of subtle stability results.

Theorem 19.4 and the following analysis are from Higham [549, 1991].

Arioli, Duff, and de Rijk [25, 1989] investigate the application of fixed precision iterative refinement to large, sparse LS problems, taking the basic solver to be the block LDL^T factorization code MA27 [329, 1982] from the Harwell Subroutine Library (applied to the augmented system); in particular, they use scaling of the form (19.17). Björck [114, 1992] determines, via an error analysis for solution of the augmented system by block LDL^T factorization, a choice of α in (19.17) that minimizes a bound on the forward error.

The idea of implementing iterative refinement with a precision that increases on each iteration (see the Notes and References to Chapter 11) can be applied to the LS problem; see Gluchowska and Smoktunowicz [453, 1990].

The use of SNE was first suggested by Kahan, in the context of iterative refinement, as explained by Golub and Wilkinson [466, 1966].

Stewart [945, 1977] discusses the problem of finding the normwise backward error for the LS problem and offers some backward perturbations that are candidates for being of minimal norm. The problem is also discussed by Higham [542, 1990]. Componentwise backward error for the LS problem has been investigated by Arioli, Duff, and de Rijk [25, 1989], Björck [113, 1991], and Higham [542, 1990].

Theorem 19.5 has been extended to the multiple right-hand side LS problem by Sun [976, 1996].

Lemma 19.6 is from a book by Kiełbasiński and Schwetlick, which has been published in German [658, 1988] and Polish [659, 1992] editions, but not in English. The lemma is their Lemma 8.2.11, and can be shown to be equivalent to a result of Stewart [943, 1977, Thin. 5.3].

Other methods for solving the LS problem not considered in this chapter include those of Peters and Wilkinson [826, 1970], Cline [215, 1973], and Plemmons [835, 1974], all of which begin by computing an LU factorization of the rectangular matrix A. Error bounds for these methods can be derived using results from this chapter and Chapters 9 and 18.

In this chapter we have not specifically treated LS problems whose coefficient matrices have rows varying greatly in norm, and we have not considered weighted LS problems $\min_x \|D(b - Ax)\|_2$, where $D = \mathrm{diag}(d_i)$. Error analysis for Householder QR factorization with column pivoting applied to badly row-scaled problems is given by Powell and Reid [840, 1969]. Methods and error analysis for weighted LS problems are given by Barlow [59, 1988], Barlow and Handy [64, 1988], Barlow and Vemulapati [66, 1992], Gulliksson and Wedin [489, 1992], Gulliksson [487, 1994], [488, 1995], and Van Loan [1042, 1985].

Another topic not considered here is the constrained LS problem, where x is required to satisfy linear equality and/or inequality constraints. Numerical methods are described by Lötstedt [713, 1984] and Golub and Van Loan [470, 1989, §12. 1], and perturbation theory is developed by Eldén [350, 1980], Lötstedt [712, 1983], and Wedin [1070, 1985].

19.9.1. LAPACK

Driver routine `xGELS` solves the full rank LS problem by Householder QR factorization. It caters for multiple right-hand sides, each of which defines a separate LS problem. Thus, `xGELS` solves $\min\{ \|B - AX\|_F : X \in \mathbb{R}^{n\times p}\}$, where $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ and $B \in \mathbb{R}^{m\times p}$. This routine does not return any error bounds, and iterative refinement is not supported for LS problems in LAPACK.

Driver routines `xGELSX` and `xGELSS` solve the rank-deficient LS problem with multiple right-hand sides, using, respectively, a complete orthogonal factorization (computed via QR factorization with column pivoting) and the SVD.

LAPACK also contains routines for solving the linearly constrained LS problem (`xGGLSE`) and a generalized form of weighted LS problem (`xGGGLM`).

Problems

19.1. Show that any solution to the LS problem $\min_x \|b - Ax\|_2$ satisfies the normal equations $A^TAx = A^Tb$. What is the geometrical interpretation of these equations?

19.2. Prove Theorem 19.3.

19.3. The pseudo-inverse $X \in \mathbb{R}^{n\times m}$ of $A \in \mathbb{R}^{m\times n}$ can be defined as the unique matrix satisfying the four *Moore–Penrose* conditions

$$\begin{array}{ll} \text{(i)}\ AXA = A, & \text{(ii)}\ XAX = X, \\ \text{(iii)}\ AX = (AX)^T, & \text{(iv)}\ XA = (XA)^T. \end{array}$$

Let $A = USV^T$ be an SVD, with $S = \mathrm{diag}(\sigma_i)$ and let $r = \mathrm{rank}(A)$. Show that $X = V\,\mathrm{diag}(\sigma_1^{-1}, \ldots, \sigma_r^{-1}. 0, \ldots, 0)U^T$ satisfies (i)-(iv) and hence is the pseudo-inverse of A. Show that $(A^+)^+ = A$.

19.4. Show that the pseudo-inverse A^+ of $A \in \mathbb{R}^{m\times n}$ solves the problem

$$\min_{X\in\mathbb{R}^{n\times m}} \|AX - I_m\|_F.$$

Is the solution unique?

19.5. Prove a result analogous to Theorem 19.3 for the MGS method, as described in §19.3.

19.6. Consider the LS problem $\min_x \|b - Ax\|_2$, where $A \in \mathbb{R}^{m\times n}$. Let $\widehat{x}$ be the computed LS solution obtained from the normal equations method and x the exact solution, and define $\widehat{r} = b - A\widehat{x}$, $r = b - Ax$. Using (19.12) and (19.13) show that a bound holds of the form

$$|r^T r - \widehat{r}^T \widehat{r}| \le c_{m,n} u \|x - \widehat{x}\|_2 \|A\|_2 \big(\|A\|_2 \|\widehat{x}\|_2 + \|b\|_2\big) + O(u^2).$$

19.7. Prove (19.18) and (19.19).

19.8. (Waldén, Karlson, and Sun [1060, 1995]) Partially complete the gap in Theorem 19.5 by evaluating $h_F(0)$ for the case "$\theta = \infty$", that is, $Db \equiv 0$.

19.9. Prove (19.21).

Chapter 20
Underdetermined Systems

I'm thinking of two numbers.
Their average is 3.
What are the numbers?

— CLEVE B. MOLER, *The World's Simplest Impossible Problem* (1990)

This problem arises in important algorithms
used in mathematical programming . . .
In these cases, B is usually very large and sparse and,
because of storage difficulties,
it is often uneconomical to store and access Q_1 *. . .*
Sometimes it has been thought that [the seminormal equations method]
could be disastrously worse than [the Q method] . . .
It is the purpose of this note to show that such algorithms are
numerically quite satisfactory.

— *C.* C. PAIGE, *An Error Analysis of a*
Method for Solving Matrix Equations (1973)

Having considered well-determined and overdetermined linear systems, we now turn to the remaining class of linear systems: those that are underdetermined.

20.1. Solution Methods

Consider the underdetermined system $Ax = b$, where $A \in \mathbb{R}^{m\times n}$ with $m \le n$. The system can be analysed using a QR factorization

$$A^T = Q \begin{bmatrix} R \\ 0 \end{bmatrix}, \tag{20.1}$$

where $Q \in \mathbb{R}^{n\times n}$ is orthogonal and $R \in \mathbb{R}^{m\times m}$ is upper triangular. (We could, alternatively, use an LQ factorization of A, but we will keep to the standard notation.) We have

$$b = Ax = [R^T \;\; 0]Q^Tx = R^Ty_1, \tag{20.2}$$

where

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = Q^Tx, \qquad y_1 \in \mathbb{R}^m.$$

If A has full rank then $y_1 = R^{-T}b$ is uniquely determined and all solutions of $Ax = b$ *are* given by

$$x = Q \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad y_2 \in \mathbb{R}^{n-m} \text{ arbitrary}.$$

The unique solution x_{LS} that minimizes $\|x\|_2$ is obtained by setting $y_2 = 0$. We have

$$\begin{aligned} x_{LS} &= Q \begin{bmatrix} R^{-T}b \\ 0 \end{bmatrix} &(20.3)\\ &= Q \begin{bmatrix} R \\ 0 \end{bmatrix} R^{-1}R^{-T}b = Q \begin{bmatrix} R \\ 0 \end{bmatrix} (R^TR)^{-1}b &(20.4)\\ &= A^T(AA^T)^{-1}b \\ &= A^+b, \end{aligned}$$

where $A^+ = A^T(AA^T)^{-1}$ is the pseudo-inverse of A. Hence x_{LS} can be characterized as $x_{LS} = A^Ty$, where y solves the *normal equations* $AA^Ty = b$.

Equation (20.3) defines one way to compute x_{LS}. We will refer to this method as the "Q method". When A is large and sparse it is desirable to avoid storing and accessing Q, which can be expensive. An alternative method with this property uses the QR factorization (20.1) but computes x_{LS} as $x_{LS} = A^Ty$, where

$$R^TRy = b \tag{20.5}$$

(cf. (20.4)). These latter equations are called the *seminormal equations* (SNE). As the "semi" denotes, however, this method does not explicitly form AA^T, which would be undesirable from the standpoint of numerical stability. Note that equations (20.5) are different from the equations $R^TRx = A^Tb$ for an overdetermined least squares (LS) problem, where $A = Q[R^T \quad 0]^T \in \mathbb{R}^{m\times n}$ with $m \geq n$, which are also called seminormal equations (see §19.6).

20.2. Perturbation Theory

A componentwise perturbation result for the minimum 2-norm solution to an underdetermined system is readily obtained.

Theorem 20.1 (Demmel and Higham). *Let $A \in \mathbb{R}^{m\times n}$ $(m \leq n)$ be of full rank and $0 \neq b \in \mathbb{R}^m$. Suppose $||A^+ \; DA||_2 < 1$ and*

$$|\Delta A| \leq \epsilon E, \quad |\Delta b| \leq \epsilon f.$$

If x and y are the minimum 2-norm solutions to $Ax = b$ and $(A + DA)y = b + Db$, respectively, then, for any monotonic norm,

$$\frac{\|x-y\|}{\|x\|} \leq \Big(\| \, |I - A^+A| \cdot E^T \cdot |A^{+^T}x| \, \| + \| \, |A^+| \cdot (f + E|x|) \, \| \Big) \frac{\epsilon}{\|x\|} + O(\epsilon^2). \tag{20.6}$$

For any Hölder p-norm, the bound is attainable to within a constant factor depending on n.

Proof. The perturbed matrix $A + DA = A(I + A^+ \; DA)$ has full rank, so we can manipulate the equation

$$y = (A + DA)^T((A + DA)(A + DA)^T)^{-1}(b + Db)$$

to obtain

$$\begin{aligned} y - x &= (I - A^+ A) \; DA^T(AA^T)^{-1}b + A^+(Db - DAx) + O(\epsilon^2) \\ &= (I - A^+A) \; DA^TA^{+T}x + A^+(Db - DAx) + O(e^2). \end{aligned} \tag{20.7}$$

The required bound follows on using absolute value inequalities and taking norms. That the bound is attained to within a constant factor depending on n for Holder p-norms is a consequence of the fact that the two vectors on the right-hand side of (20.7) are orthogonal. □

Two special cases are worth noting, for later use. We will use the equality $\|I - A^+A\|_2 = \min\{1, n - m\}$, which can be derived by consideration of the QR factorization (20.1), for example. If $E = |A|H$, where H is a given nonnegative matrix, and $f = |b|$, then we can put (20.6) in the form

$$\frac{\|x-y\|_2}{\|x\|_2} \le \min\{3, n-m+2\} \max\{\, \|H\|_2, 1\,\} \operatorname{cond}_2(A)\epsilon + O(\epsilon^2), \qquad (20.8)$$

where

$$\operatorname{cond}_2(A) = \|\, |A^+||A| \,\|_2.$$

Note that $\operatorname{cond}_2(A)$ is independent of the row scaling of A ($\operatorname{cond}_2(DA) = \operatorname{cond}_2(A)$ for nonsingular diagonal D). If $E = \|A\|_2 e_m e_n^T$ and $f = \|b\|_2 e_m$, where e_m denotes the m-dimensional vector of 1s, then

$$\frac{\|x-y\|_2}{\|x\|_2} \le \min\{3, n-m+2\}(mn)^{1/2}\kappa_2(A)\epsilon + O(\epsilon^2). \qquad (20.9)$$

The following analogue of Lemma 19.6 will be needed for the error analysis in the next section, It says that if we perturb the two occurrences of A in the normal equations $AA^Tx = b$ differently, then the solution of the perturbed system is the solution of normal equations in which there is only one perturbation of A and, moreover, this single perturbation is no larger, in the normwise or componentwise sense, than the two perturbations we started with.

Lemma 20.2 (Kiełbasiński and Schwetlick). *Let $A \in \mathbb{R}^{m\times n}$ ($m \le n$) be of full rank and suppose*

$$(A + \Delta A_1)\overline{x} = b, \qquad \overline{x} = (A + \Delta A_2)^T\overline{y}.$$

Assume that $3\max(\|A^+\Delta A_1\|_2, \|A^+\Delta A_2\|_2) < 1$. *Then there is a vector $\widetilde{y}$ and a perturbation ΔA with*

$$\Delta A = \Delta A_1 G_1 + \Delta A_2 G_2, \qquad G_1^T = G_1 = G_1^2, \qquad G_1 + G_2 = I,$$

such that

$$(A + \Delta A)\overline{x} = b, \qquad \overline{x} = (A + \Delta A)^T\widetilde{y},$$

that is, $\overline{x}$ is the minimum 2-norm solution to $(A + \Delta A)x = b$.

Proof. The proof is similar to that of Lemma 19.6, but differs in some details. If $\overline{x} = (A + \Delta A_2)^T\overline{y} = 0$ we take $\Delta A = \Delta A_2$. otherwise, we set

$$\Delta A := \Delta A_1 P + \Delta A_2(I - P) =: \Delta A_2 + HP,$$

where $P = \overline{x}\overline{x}^T/\overline{x}^T\overline{x}$ and $H = DA_1 - DA_2$. We have

$$(A + \Delta A)^T\overline{y} = (A + \Delta A_2 + HP)^T\overline{y} = \beta\overline{x},$$

where $b = 1 + \overline{x}^T H^T\overline{y}/\overline{x}^T\overline{x}$, which shows that we need to set $\widetilde{y} = \beta^{-1}\overline{y}$. To check that $(A + DA)\overline{x} = b$, we evaluate

$$\begin{aligned}(A + \Delta A)\overline{x} &= (A + \Delta A_1 + H(P - I))\overline{x}\\ &= (A + \Delta A_1)\overline{x} = b,\end{aligned}$$

as required. The vector $\widetilde{y}$ is undefined if $b = 0$. But

$$\begin{aligned}\beta &= 1 + \frac{\overline{x}^T H^T\overline{y}}{\overline{x}^T\overline{x}} = 1 + \frac{\overline{x}^T H^T(A + \Delta A_2)^{+T}\overline{x}}{\overline{x}^T\overline{x}}\\ &\geq 1 - \|(A + \Delta A_2)^+ H\|_2\\ &\geq 1 - \frac{\|A^+ H\|_2}{1 - \|A^+\Delta A_2\|_2} \qquad (\text{if } \|A^+\Delta A_2\|_2 < 1)\\ &\geq 1 - \frac{\big(\|A^+\Delta A_1\|_2 + \|A^+\Delta A_2\|_2\big)}{1 - \|A^+\Delta A_2\|_2},\end{aligned}$$

which is positive if 3 max $(\|A^+ DA_1\|_2, \|A^+ DA_2\|_2) < 1$. □

Note that in Lemma 20.2 we have the normwise bound

$$\|\Delta A\|_p \leq \big(\|\Delta A_1\|_p^2 + \|\Delta A_2\|_p^2\big)^{1/2}, \quad p = 2, F.$$

20.3. Error Analysis

We now consider the stability of the Q method and the SNE method. For both methods we assume that the QR factorization is computed using Householder or Givens transformations.

Before presenting the results we define a measure of stability. The componentwise backward error for a minimum-norm underdetermined system $Ax = b$ is defined as

$$\begin{aligned}\overline{\omega}_{E,f}(y) := \min\{\,\epsilon : \exists\ \Delta A \in \mathbb{R}^{m\times n}, \Delta b \in \mathbb{R}^m \text{ with } |\Delta A| \leq \epsilon E,\ |\Delta b| \leq \epsilon f\\ \text{s.t. } y \text{ is the min. norm solution to } (A + DA)y = b + Db\ \},\end{aligned}$$

Note the requirement in this definition that y be the minimum norm solution; the usual componentwise backward error $\omega_{E,f}(y)$ (see (7.6)) is a generally smaller quantity. Let us say that a method is row-wise *backward stable* if it produces a computed solution $\widehat{x}$ for which the componentwise backward error

$\overline{\omega}_{E,f}$ is of order u, where $E = |A|e_m e_n^T$ and $f = |b|$. This condition requires that $\widehat{x}$ solve a perturbed minimum norm problem in which the perturbations to the ith row of A are small compared with the norm of that row (similarly for b); cf. the discussion of componentwise backward errors in §7.2.

Theorem 20.3. *Let $A \in \mathbb{R}^{m\times n}$ with $\operatorname{rank}(A) = m \le n$, and assume that a rendition of the form $\operatorname{cond}_2(A)mn\gamma_{cn} < 1$ holds. Suppose the underdetermined system $Ax = b$ is solved in the minimum 2-norm sense using the Q method. Then the computed solution $\widehat{x}$ is the minimum 2-norm solution to $(A+\Delta A)x = b$, where*

$$\|\Delta A\|_F \le m\gamma_{cn}\|A\|_F$$

and

$$|\Delta A| \le mn\gamma_{cn}|A|G, \qquad \|G\|_F = 1.$$

Proof. The Q method solves the triangular system $R^T y_1 = b$ and then forms $x = Q[y_1^T,\ 0]^T$. Assuming the use of Householder QR factorization, from Theorem 18.4 we have that

$$(A+\Delta A_0)^T = Q\begin{bmatrix}\widehat{R}\\ 0\end{bmatrix},$$

for some orthogonal matrix Q, where $\|\Delta A_0\|_F \le m\gamma_{cn}\|A\|_F$ and $|\Delta A_0^T| \le mn\gamma_{cn}G_0^T|A^T|$ with $\|G_0\|_F = 1$. The computed $\widehat{y}_1$ satisfies

$$(\widehat{R}+\Delta\widehat{R})^T\widehat{y}_1 = b, \qquad |\Delta\widehat{R}| \le \gamma_m|\widehat{R}|.$$

From Lemma 18.3, the computed solution $\widehat{x}$ satisfies

$$\widehat{x} = (Q+\Delta Q)\begin{bmatrix}\widehat{y}_1\\ 0\end{bmatrix}, \qquad \|\Delta Q\|_F \le m\gamma_{cn}. \tag{20.10}$$

We now rewrite the latter two equations in such a way that we can apply Lemma 20.2:

$$(A+\Delta A_1)\widehat{x} := \begin{bmatrix}(\widehat{R}+\Delta\widehat{R})^T & 0\end{bmatrix}(Q+\Delta Q)^{-1}\cdot(Q+\Delta Q)\begin{bmatrix}\widehat{y}_1\\ 0\end{bmatrix} = b,$$

$$\widehat{x} = (Q+\Delta Q)\begin{bmatrix}\widehat{R}\\ 0\end{bmatrix}\cdot\widehat{R}^{-1}\widehat{y}_1 =: (A+\Delta A_2)^T\overline{y}.$$

It is straightforward to show that

$$|\Delta A_i| \le mn\gamma'_{cn}|A|G_i, \qquad \|G_i\|_F = 1,\ i = 1{:}2,$$

and $\|\Delta A_i\|_F \le m\gamma'\|A\|_F$, $i = 1{:}2$. The result follows on invocation of Lemma 20.2. $\square$

Theorem 20.3 says that the Q method is row-wise backward stable. This is not altogether surprising, since (Householder or Givens) QR factorization for the LS problem enjoys an analogous backward stability result (Theorem 19.3), albeit without the restriction of a minimum norm solution. Applying (20.8) to Theorem 20.3 we obtain the forward error bound

$$\frac{\|\widehat{x}-x\|_2}{\|x\|_2} \le mn\gamma'_{cn}\,\mathrm{cond}_2(A) + O(u^2). \qquad (20.11)$$

The same form of forward error bound (20.11) can be derived for the SNE method as for the Q method [292, 1993]. However, it is not possible to obtain a result analogous to Theorem 20.3, nor even to obtain a residual bound of the form $\|b - A\widehat{x}\|_2 \le c_{m,n}u\|A\|_2\|\widehat{x}\|_2$ (which would imply that $\widehat{x}$ solved a nearby system, though $\widehat{x}$ would not necessarily be the minimum norm solution). The method of solution guarantees only that the seminormal equations themselves have a small residual. Thus, as in the context of overdetermined LS problems, the SNE method is not backward stable. A possible way to improve the stability is by iterative refinement, as shown in [292, 1993].

Note that the forward error bound (20. 11) is independent of the row scaling of A, since $\mathrm{cond}_2(A)$ is. The bound is therefore potentially much smaller than the bound

$$\frac{\|x-\widehat{x}\|_2}{\|x\|_2} \le c_{m,n}u\kappa_2(A) + O(u^2),$$

obtained by Paige [813, 1973] for the SNE method and by Jennings and Osborne [614, 1974] and Arioli and Laratta [26, 1985, Thin. 4] for the Q method.

Finally, we mention an alternative version of the Q method that is based on the modified Gram–Schmidt (MGS) method. The obvious approach is to compute the QR factorization $AT = QR$ using MGS ($Q \in \mathbb{R}^{n\times m}$, $R \in \mathbb{R}^{m\times m}$) solve $R^Ty = b$, and then form $x = Qy$. Since Q is provided explicitly by the MGS method, the final stage is a full matrix–vector multiplication, unlike for the Householder method. However, because the computed Q may depart from orthonormality, this method is unstable in the form described. The formation of $x = Qy$ should instead be done as follows:

```
$x^{(n)} = 0$
for $k = n: -1: 1$
    $\alpha_k = q_k^T x^{(k)}$
    $x^{(k-1)} = x^{(k)} - (\alpha_k - y_k)q_k$
end
$x = x^{(0)}$
```

The recurrence can be written as $x^{(k-l)} = x^{(k)} + y_kq_k - (q_k{}^Tx^{(k)})q_k$, and the last term is zero in exact arithmetic if the q_k are mutually orthogonal. In finite precision arithmetic this correction term has the "magical" effect of making

Table 20.1. *Backward errors for underdetermzned Vandermonde system.*

	$\eta_{A,b}(\widehat{x})$
Householder QR	9.76×10^{-18}
MGS with $x := Qy$	4.10×10^{-4}
MGS with x formed stably (see text)	2.25×10^{-17}
SNE method (using Householder QR)	1.99×10^{-4}

the algorithm stable, in the sense that it satisfies essentially the same result as the Q method in Theorem 20.3; see Björck and Paige [120, 1994].

A numerical example is instructive. Take the 20 x 30 Vandermonde matrix $A = (p_i^{j-1})$, where the p_i are equally spaced on [0, 1], and let b have elements equally spaced on [0, 1]. The condition number $\kappa_2(A) = 4.35 \times 10^{14}$. The (standard) normwise backward errors in the 2-norm are shown in Table 20.1. For A^T, the $\widehat{Q}$ supplied by MGS satisfies $\|\widehat{Q}^T\widehat{Q} - I\|_2 = 1.41 \times 10^{-3}$, which explains the instability of the "obvious" MGS solver.

20.4. Notes and References

The seminormal equations method was suggested by Gill and Murray [442, 1973] and Saunders [894, 1972]. Other methods for obtaining minimal 2-norm solutions of underdetermined systems are surveyed by Cline and Plemmons [219, 1976].

Theorem 20.1 is from Demmel and Higham [292, 1993]. The bound (20.9) is well known; it follows from Wedin's original version of our Theorem 19.1, which applies to minimum 2-norm underdetermined problems as well as LS problems.

Theorem 20.3 is new. Demmel and Higham [292, 1993] prove the weaker result that $\widehat{x}$ from the Q method is *very close to* a vector $\overline{x}$ that satisfies the criterion for row-wise backward stability, and Lawson and Hanson [695, 1974, Thin. 16. 18] give a corresponding result in which $\overline{x}$ satisfies the criterion for general normwise backward stability. The key to showing actual backward stability is the use of Kiełbasiński and Schwetlick's lemma, which is a modification of Lemma 8.2.11 in [658, 1988] and [659, 1992] (our Lemma 19.6). Demmel and Higham [292, 1993] also give error analysis for the seminormal equations method.

The new MGS algorithm for solving the minimum norm problem was first suggested by Björck and Paige [119, 1992]; see also Björck [115, 1994].

Arioli and Laratta [27, 1986] give error analysis of QR factorization methods for solving the general problem $\min\{\, \|x - c\|_2 : Ax = b \,\}$, where $A \in \mathbb{R}^{m\times n}$

with $m \leq n$.

20.4.1. LAPACK

The same routines that solve the (overdetermined) LS problem also solve underdetermined systems for the solution of minimal 2-norm. Thus, xGELS solves a full rank underdetermined system with multiple right-hand sides by the Q method. Routines xGELSX and xGELSS solve rank-deficient problems with multiple right-hand sides, using, respectively, a complete orthogonal factorization (computed via QR factorization with column pivoting) and the singular value decomposition.

Problems

20.1. (Björck [114, 1992]) Show that the system

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} y \\ x \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix} \tag{20.12}$$

characterizes the solution to the following generalizations of the LS problem and the problem of finding the minimum norm solution to an underdetermined system:

$$\min_x \|b - Ax\|_2^2 + 2c^T x, \tag{20.13}$$

$$\min_y \|y - b\|_2 \quad \text{subject to} \quad A^T y = c. \tag{20.14}$$

20.2. (RESEARCH PROBLEM) Find a formula for the backward error of an arbitrary approximation to the minimum 2-norm solution of an underdetermined system. That is, for $A \in \mathbb{R}^{m \times n}$ with $\operatorname{rank}(A) = m \leq n$, find

$$m(y) := \min\{\, e : y \text{ is the minimum 2-norm solution to } (A + DA)y = b + Db, \\ \text{where } \|DA\|_2 \leq e\|A\|_2,\ \|Db\|_2 \leq e\|b\|_2 \,\}.$$

Previous Home Next

Chapter 21

Vandermonde Systems

We began, 25 years ago, to take up [the conditioning of]
the class of Vandermonde matrices.
The original motivation came from unpleasant experiences with the
computation of Gauss type quadrature rules from the
moments of the underlying weight function.

— WALTER GAUTSCHI, *How (Un)stable are Vandermonde Systems?* (1990)

Extreme ill-conditioning of the [Vandermonde] linear systems
will eventually manifest itself as n increases by yielding
an error curve which is not sufficiently levelled on the current reference . . .
or more seriously fails to have the correct number of sign changes.

— M. ALMACANY, C. B. DUNHAM, and J. WILLIAMS,
Discrete Chebyshev Approximation by Interpolating Rationals (1984)

A Vandermonde matrix is defined in terms of scalars $\alpha_0, \alpha_1, \ldots, \alpha_n \in \mathbb{C}$ by

$$V = V(\alpha_0, \alpha_1, \ldots, \alpha_n) = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ \alpha_0 & \alpha_1 & \ldots & \alpha_n \\ \vdots & \vdots & & \vdots \\ \alpha_0^n & \alpha_1^n & \ldots & \alpha_n^n \end{bmatrix} \in \mathbb{C}^{(n+1)\times(n+1)}.$$

Vandermonde matrices play an important role in various problems, such as in polynomial interpolation. Suppose we wish to find the polynomial $p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0$ that interpolates to the data $(\alpha_i, f_i)_{i=0}^n$, for distinct points α_i, that is, $p_n(\alpha_i) = f_i$, $i = 0{:}n$. Then the desired coefficient vector $a = [a_0, a_1, \ldots, a_n]^T$ is the solution of the dual Vandermonde system

$$V^T a = f \quad \text{(dual)}.$$

The primal system

$$Vx = b \quad \text{(primal)}$$

represents a moment problem, which arises, for example, when determining the weights for a quadrature rule: given moments b_i find weights x_i such that $\sum_{j=0}^n x_j \alpha_j^i = b_i$, $i = 0{:}n$.

Because a Vandermonde matrix depends on only n+ 1 parameters and has a great deal of structure, it is possible to perform standard computations with reduced complexity. The easiest algorithm to derive is for matrix inversion.

21.1. Matrix Inversion

Assume that V is nonsingular and let $V^{-1} = W = (w_{ij})_{i,j=0}^n$. The ith row of the equation $WV = I$ may be written

$$\sum_{j=0}^n w_{ij}\alpha_k^j = \delta_{ik}, \quad k = 0{:}n.$$

These equations specify a fundamental interpolation problem that is solved by the Lagrange basis polynomial:

$$\sum_{j=0}^n w_{ij}x^j = \prod_{\substack{k=0\\k\neq i}}^n \left(\frac{x-\alpha_k}{\alpha_i-\alpha_k}\right) =: l_i(x). \tag{21.1}$$

The inversion problem is now reduced to finding the coefficients of $l_i(x)$. It is clear from (21.1) that V is nonsingular iff the α_i are distinct. It also follows from (21. 1) that V^{-1} is given explicitly by

$$w_{ij} = \frac{(-1)^{n-j}\sigma_{n-j}(\alpha_0, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n)}{\prod\limits_{\substack{k=0\\k\neq i}}^n (\alpha_i - \alpha_k)}, \tag{21.2}$$

where $s_k(y_1, \ldots, y_n)$ denotes the sum of all distinct products of k of the arguments $y_1, \ldots, y_n$ (that is σ_k is the kth elementary symmetric function). An efficient way to find the w_{ij} is first to form the master polynomial

$$\phi(x) = \prod_{k=0}^{n} (x - \alpha_k) =: \sum_{i=0}^{n+1} a_i x^i,$$

and then to recover each Lagrange polynomial by synthetic division:

$$q_i(x) = \phi(x)/(x - \alpha_i),$$
$$l_i(x) = q_i(x)/q_i(\alpha_i).$$

The scalars $q_i(\alpha_i)$ can be computed by Homer's rule as the coefficients of q_i are formed.

Algorithm 21.1. Given distinct scalars $\alpha_0, \alpha_1, \ldots, \alpha_n \in \mathbb{C}$ this algorithm computes $W = (w_{ij})_{i,j=0}^{n} = V(\alpha_0, \alpha_1, \ldots, \alpha_n)^{-1}$.

```
% Stage 1: Construct the master polynomial.
$a_0 = -\alpha_0$; $a_1 = 1$
for $k = 1:n$
    $a_{k+1} = 1$
    for $j = k: -1: 1$
        $a_j = a_{j-1} - \alpha_k a_j$
    end
    $a_0 = -\alpha_k a_0$
end

% Stage 2: Synthetic division.
for $i = 0:n$
    $w_{in} = 1$; $s = 1$
    for $j = n - 1: -1:0$
        $w_{ij} = a_{j+1} + \alpha_i w_{i,j+1}$
        $s = \alpha_i s + w_{ij}$
    end
    $w(i, :) = w(i, :)/s$
end
```

Cost: $6n^2$ flops.

The $O(n^2)$ complexity is optimal, since the algorithm has n^2 output values, each of which must partake in at least one operation.

Vandermonde matrices have the deserved reputation of being ill conditioned. The ill conditioning is a consequence of the monomials being a poor

Table 21.1. *Bounds and estimates for* $\kappa_\infty(V_n)$.

	α_i	Bound or estimate	Reference
(VI):	$1/i+1$	$\kappa_\infty(V_n) > n^{n+1}$	[428, 1990]
(V2):	arbitrary	$\kappa_2(V_n) \geq n^{-1/2}2^{n-2}$	[1035, 1994]
(V3):	$\alpha_i \geq 0$	$\kappa_\infty(V_n) > 2^{n-1}$ $(n > 1)$	[429, 1988]
(V4):	equispaced [0,1]	$\kappa_\infty(V_n) \sim (4\pi)^{-1}\sqrt{2}\, 8^n$	[428, 1990]
(V5):	equispaced [– 1, 1]	$\kappa_\infty(V_n) \sim \pi^{-1}e^{-\pi/4}(3.1)^n$	[425, 1975]
(V6):	Chebyshev nodes [–1, 1]	$\kappa_\infty(V_n) \sim \frac{3^{3/4}}{4}(1+\sqrt{2})^n$	[425, 1975]
(V7):	roots of unity	$\kappa_2(V_n) = 1$	well known.

basis for the polynomials on the real line. A variety of bounds for the condition number of a Vandermonde matrix have been derived by Gautschi and his co-authors. Let $V_n = V(\alpha_0, \alpha_1, \dots, \alpha_{n-1}) \in \mathbb{C}^{n\times n}$. For arbitrary distinct points α_i,

$$\max_i \prod_{j\neq i} \frac{\max(1,|\alpha_j|)}{|\alpha_i - \alpha_j|} \leq \|V_n^{-1}\|_\infty \leq \max_i \prod_{j\neq i} \frac{1+|\alpha_j|}{|\alpha_i - \alpha_j|}, \tag{21.3}$$

with equality on the right when $\alpha_j = |\alpha_j|e^{i\theta}$ for all j with a fixed q (in particular, when $\alpha_j \geq 0$ for all j) [424, 1962], [426, 1978]. Note that the upper and lower bounds differ by at most a factor 2^{n-1}. More specific bounds are given in Table 21.1, on which we now comment.

Bound (V1) and estimate (V4) follow from (21.3). The condition number for the harmonic points $1/(i + 1)$ grows faster than $n!$; by contrast, the condition numbers of the notoriously ill-conditioned Hilbert and Pascal matrices grow only exponentially (see §26.1 and §26.4). For *any* choice of points the rate of growth is at least exponential (V2), and this rate is achieved for points equally spaced on [0, 1]. For points equally spaced on [– 1, 1], the condition number grows at a slower exponential rate than that for [0, 1], and the growth rate is slower still for the zeros of the nth degree Chebyshev polynomial (V6). For one set of points the Vandermonde matrix is perfectly conditioned: the roots of unity, for which $V_n/\sqrt{n}$ is unitary.

21.2. Primal and Dual Systems

The standard Vandermonde matrix can be generalized in at least two ways: by allowing confluence of the points α_i and by replacing the monomials by other polynomials. An example of a confluent Vandermonde matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ \alpha_0 & 1 & 0 & \alpha_1 & 1 \\ \alpha_0^2 & 2\alpha_0 & 2 & \alpha_1^2 & 2\alpha_1 \\ \alpha_0^3 & 3\alpha_0^2 & 6\alpha_0 & \alpha_1^3 & 3\alpha_i^2 \\ \alpha_0^4 & 4\alpha_0^3 & 12\alpha_0^2 & \alpha_1^4 & 4\alpha_i^3 \end{bmatrix}. \tag{21.4}$$

The second, third, and fifth columns are obtained by "differentiating" the previous column. The transpose of a confluent Vandermonde matrix arises in Hermite interpolation; it is nonsingular if the points corresponding to the "nonconfluent columns" are distinct.

A *Vandermonde-like matrix* is defined by $P = (p_i(\alpha_j))_{i,j=0}^n$, where p_i is a polynomial of degree *i*. The case of practical interest is where the pi satisfy a three-term recurrence relation. In the rest of this chapter we will assume that the p_i do satisfy a three-term recurrence relation. A particular application is the solution of certain discrete Chebyshev approximation problems [11, 1984]. Incorporating confluence, we obtain a *confluent Vandermonde-like matrix,* defined by

$$P = P(\alpha_0, \alpha_1, \ldots, \alpha_n) = [q_0(\alpha_0), q_1(\alpha_1), \ldots. q_n(\alpha_n)]\ \mathbb{C}^{(n+1)\times(n+1)},$$

where the α_i are ordered so that equal points are contiguous, that is,

$$\alpha_i = \alpha_j \quad (i < j) \quad \Rightarrow \quad \alpha_i = \alpha_{i+1} = \cdots = \alpha_j, \tag{21.5}$$

and the vectors $q_j(x)$ are defined recursively by

$$q_j(x) = \begin{cases} [p_0(x),\ p_1(x),\ \ldots,\ p_n(x)]^T, & \text{if } j = 0 \text{ or } \alpha_j \neq \alpha_{j-1}, \\ \dfrac{d}{dx} q_{j-1}(x), & \text{otherwise.} \end{cases}$$

For all polynomials and points, *P* is nonsingular; this follows from the derivation of the algorithms below. One reason for the interest in Vandermonde-like matrices is that for certain polynomials they tend to be better conditioned than Vandermonde matrices (see, for example, Problem 21.5). Gautschi [427, 1983] derives bounds for the condition numbers of Vandermonde-like matrices.

Fast algorithms for solving the confluent Vandermonde-like primal and dual systems $Px = b$ and $P^Ta = f$ can be derived under the assumption that the $p_j(x)$ satisfy the three-term recurrence relation

$$p_{j+1}(x) = \theta_j(x - \beta_j)p_j(x) - \gamma_j p_{j-1}(x), \quad j \geq 1, \tag{21.6a}$$

$$p_0(x) = 1, \qquad p_1(x) = \theta_0(x - \beta_0)p_0(x), \tag{21.6b}$$

where $\theta_j \neq 0$ for all *j*. Note that in this chapter γ_i denotes a constant in the recurrence relation and not $iu/(1 - iu)$ as elsewhere in the book. The latter notation is not used in this chapter.

The algorithms exploit the connection with interpolation. Denote by $r(i) \geq 0$ the smallest integer for which $\alpha_i = \alpha_{i-1} = \cdot \cdot \cdot = \alpha_{r(i)}$. Considering first the dual system $P^T a = f$, we note that

$$\psi(x) = \sum_{i=0}^{n} a_i p_i(x) \tag{21.7}$$

satisfies

$$\psi^{(i-r(i))}(\alpha_i) = f_i, \qquad i = 0{:}n.$$

Thus y is a Hermite interpolating polynomial for the data $\{\alpha_i, f_i\}$, and our task is to obtain its representation in terms of the basis $\{p_i(x)\}_{i=0}^{n}$. As a first step we construct the divided difference form of y,

$$\psi(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1} (x - \alpha_j). \tag{21.8}$$

The (confluent) divided differences $c_i = f[\alpha_0, \alpha_1, \ldots \alpha_i]$ may be generated using the recurrence relation

$$f[\alpha_{j-k-1}, \ldots, \alpha_j] = \begin{cases} \dfrac{f[\alpha_{j-k}, \ldots, \alpha_j] - f[\alpha_{j-k-1}, \ldots, \alpha_{j-1}]}{\alpha_j - \alpha_{j-k-1}}, & \alpha_j \neq \alpha_{j-k-1}, \\[2ex] \dfrac{f_{r(j)+k+1}}{(k+1)!}, & \alpha_j = \alpha_{j-k-1}. \end{cases} \tag{21.9}$$

Now we need to generate the a_i in (21.7) from the ci in (21.8). The idea is to expand (21.8) using nested multiplication and use the recurrence relations (21.6) to express the results as a linear combination of the p_j. Define

$$q_n(x) = c_n, \tag{21.10}$$

$$q_k(x) = (x - \alpha_k) q_{k+1}(x) + c_k, \quad k = n-1{:}-1{:}0, \tag{21.11}$$

from which $q_0(x) = y(x)$. Let

$$q_k(x) = \sum_{j=0}^{n-k} a_{k+j}^{(k)} p_j(x). \tag{21.12}$$

To obtain recurrences for the coefficients $a_j^{(k)}$ we expand the right-hand side of (21.11), giving

$$q_k(x) = (x - \alpha_k) \sum_{j=0}^{n-k-1} a_{k+j+1}^{(k+1)} p_j(x) + c_k.$$

Using the relations, from (21.6),

$$
\begin{aligned}
xp_0(x) &= \frac{1}{\theta_0}p_1(x) + \beta_0,\\
xp_j(x) &= \frac{1}{\theta_j}\big(p_{j+1}(x) + \gamma_j p_{j-1}(x)\big) + \beta_j p_j(x), \quad j \ge 1,
\end{aligned}
$$

we obtain, for $k = 0{:}n-2$,

$$
\begin{aligned}
q_k(x) &= a_{k+1}^{(k+1)}\left(\frac{1}{\theta_0}p_1(x) + \beta_0\right)\\
&\quad + \sum_{j=1}^{n-k-1} a_{k+j+1}^{(k+1)}\left(\frac{1}{\theta_j}\big(p_{j+1}(x) + \gamma_j p_{j-1}(x)\big) + \beta_j p_j(x)\right)\\
&\quad - \alpha_k \sum_{j=0}^{n-k-1} a_{k+j+1}^{(k+1)} p_j(x) + c_k\\
&= c_k + (\beta_0 - \alpha_k)a_{k+1}^{(k+1)} + \frac{\gamma_1}{\theta_1}a_{k+2}^{(k+1)}\\
&\quad + \sum_{j=1}^{n-k-2}\left(\frac{1}{\theta_{j-1}}a_{k+j}^{(k+1)} + (\beta_j - \alpha_k)a_{k+j+1}^{(k+1)} + \frac{\gamma_{j+1}}{\theta_{j+1}}a_{k+j+2}^{(k+1)}\right)p.\\
&\quad + \left(\frac{1}{\theta_{n-k-2}}a_{n-1}^{(k+1)} + (\beta_{n-k-1} - \alpha_k)a_n^{(k+1)}\right)p_{n-k-1}(x)\\
&\quad + \frac{1}{\theta_{n-k-1}}a_n^{(k+1)}p_{n-k}(x),
\end{aligned}
\tag{21.13}
$$

in which the empty summation is defined to be zero. For the special case $k = n-1$ we have

$$
q_{n-1}(x) = c_{n-1} + (\beta_0 - \alpha_{n-1})a_n^{(n)} + \frac{1}{\theta_0}a_n^{(n)}p_1(x). \tag{21.14}
$$

Recurrences for the coefficients $a_j^{(k)}$, $j = k{:}n$, in terms of $a_j^{(k+1)}$, $j = k+1{:}\,n$, follow immediately by comparing (21.12) with (21.13) and (21.14).

In the following algorithm, stage I computes the confluent divided differences and stage II implements the recurrences derived above.

Algorithm 21.2 (dual, $P^Ta = f$). Given parameters $\{\theta_j, \beta_j, \gamma_j\}_{j=0}^{n-1}$, a vector f, and points $\alpha(0{:}n) \in \mathbb{C}^{n+1}$ satisfying (21 .5), this algorithm solves the dual confluent Vandermonde-like system $P^Ta = f$.

```
% Stage I:
Set $c = f$
```

```
for $k = 0:n-1$
    $clast = c_k$
    for $j = k+1:n$
        if $\alpha_j \quad \alpha_{j-k-1}$ then
            $c_j = c_j/(k+1)$
        else
            $temp = c_j$
            $c_j = (c_j - clast)/(\alpha_j - \alpha_{j-k-1})$
            $clast = temp$
        end
    end
end

% Stage II:
Set $a = c$
$a_{n-1} = a_{n-1} + (\beta_0 - \alpha_{n-1})a_n$
$a_n = a_n/\theta_0$
for $k = n-2:-1:0$
    $a_k = a_k + (\beta_0 - \alpha_k)a_{k+1} + (\gamma_1/\theta_1)a_{k+2}$
    for $j = 1:n-k-2$
        $a_{k+j} = a_{k+j}/\theta_{j-1} + (\beta_j - \alpha_k)a_{k+j+1} + (\gamma_{j+1}/\theta_{j+1})a_{k+j+2}$
    end
    $a_{n-1} = a_{n-1}/\theta_{n-k-2} + (\beta_{n-k-1} - \alpha_k)a_n$
    $a_n = a_n/q_{n-k-1}$
end
```

Assuming that the values γ_j/θ_j are given (note that γ_j appears only in the terms γ_j/θ_j), the computational cost of Algorithm 21.2 is at most $9n^2/2$ flops. The vectors c and a have been used for clarity; in fact both can be replaced by f, so that the right-hand side is transformed into the solution without using any extra storage.

Values of θ_j, β_j, γ_j for some polynomials of interest are collected in Table 21.2.

The key to deriving a corresponding algorithm for solving the primal system is to recognize that Algorithm 21.2 implicitly computes a factorization of P^{-T} into the product of $2n$ triangular matrices. In the rest of this chapter we adopt the convention that the subscripts of all vectors and matrices run from 0 to n. In stage I, letting $c^{(k)}$ denote the vector c at the start of the kth iteration of the outer loop, we have

$$c^{(0)} = f, \qquad c^{(k+1)} = L_k c^{(k)}, \quad k = 0:n-1. \tag{21.15}$$

The matrix L_k is lower triangular and agrees with the identity matrix in rows

Table 21.2. *Parameters in the three-term recurrence* (21.6).

Polynomial	θ_j	β_j	γ_j	
Monomials	1	0	0	
Chebyshev	2^*	0	1	$^*\theta_0 = 1$
Legendre*	$\frac{2j+1}{j+1}$	0	$\frac{j}{j+1}$	$^*p_j(1) = 1$
Hermite	2	0	$2j$	
Laguerre	$-\frac{1}{j+1}$	$2j+1$	$\frac{j}{j+1}$	

0 to k. The remaining rows can redescribed, for $k + 1 \le j \le n$, by

$$e_j^T L_k = \begin{cases} e_j^T/(k+1), & \text{if } \alpha_j = \alpha_{j-k-1}, \\ (e_j^T - e_s^T)/(\alpha_j - \alpha_{j-k-1}), & \text{some } s < j, \text{ otherwise,} \end{cases}$$

where e_j is column j of the identity matrix. Similarly, stage II can be expressed as

$$a^{(n)} = c^{(n)}, \qquad a^{(k)} = U_k a^{(k+1)}, \qquad k = n-1:-1:0. \tag{21.16}$$

The matrix U_k is upper triangular, it agrees with the identity matrix in rows 0 to $k - 1$, and it has zeros everywhere above the first two superdiagonals.

From (21.15) and (21.16) we see that the overall effect of Algorithm 21.2 is to evaluate, step-by-step, the product

$$a = U_0 \ldots U_{n-1} L_{n-1} \ldots L_0 f \equiv P^{-T} f. \tag{21.17}$$

Taking the transpose of this product we obtain a representation of P^{-1}, from which it is easy to write down an algorithm for computing $x = P^{-1}b$.

Algorithm 21.3 (primal, $Px = b$). Given parameters $\{\theta_j, \beta_j, \gamma_j\}_{j=0}^{n-1}$, a vector b, and points $\alpha(0{:}n) \in \mathbb{C}^{n+1}$ satisfying (21.5), this algorithm solves the primal confluent Vandermonde-like system $Px = b$.

```
% Stage I:
Set $d = b$
for $k = 0:n-2$
    for $j = n-k:-1:2$
        $d_{k+j} = (\gamma_{j-1}/\theta_{j-1})d_{k+j-2} + (\beta_{j-1} - \alpha_k)d_{k+j-1} + d_{k+j}/\theta_{j-1}$
    end
    $d_{k+1} = (\beta_0 - \alpha_k)d_k + d_{k+1}/\theta_0$
end
$d_n = (\beta_0 - \alpha_{n-1})d_{n-1} + d_n/\theta_0$
```

```
% Stage II:
Set $x = d$
for $k = n - 1: -1:0$
    $xlast = 0$
    for $j = n: -1 :k + 1$
        if $\alpha_j = \alpha_{j-k-1}$ then
            $x_j = x_j/(k + 1)$
        else
            $temp = x_j/(\alpha_j - \alpha_{j-k-1})$
            $x_j = temp - xlast$
            $xlast = temp$
        end
    end
    $x_k = x_k - xlast$
end
```

Algorithm 21.3 has, by construction, the same operation count as Algorithm 21.2.

21.3. Stability

Algorithms 21.2 and 21.3 have interesting stability properties. Depending on the problem parameters, the algorithms can range from being very stable (in either a backward or forward sense) to very unstable.

When the pa are the monomials and the points α_i are distinct, the algorithms reduce to those of Björck and Pereyra [121, 1970]. Björck and Pereyra found that for the system $Vx = b$ with $\alpha_i = 1/(i + 3)$, $b_i = 2^{-i}$, $n = 9$, and on a computer with $u \approx 10^{-16}$,

$$\kappa_\infty(V) = 9 \times 10^{13}, \quad \max_i \frac{|x_i - \widehat{x}_i|}{|x_i|} = 5u.$$

Thus the computed solution has a tiny componentwise relative error, despite the extreme ill condition of V. Björck and Pereyra comment "It seems as if at least some problems connected with Vandermonde systems, which traditionally have been considered too ill-conditioned to be attacked, actually can be solved with good precision." This high accuracy can be explained with the aid of the error analysis below.

The analysis can be kept quite short by exploiting the interpretation of the algorithms in terms of matrix–vector products. Because of the inherent duality between Algorithms 21.2 and 21.3, any result for one has an analogue for the other, so we will consider only Algorithm 21.2.

The analysis that follows is based on the model (2.4), and so is valid only for machines with a guard digit. With the no-guard-digit model (2.6) the bounds become weaker and more complicated, because of the importance of terms $fl(\alpha_j - \alpha_{j-k-1})$ in the analysis.

21.3.1. Forward Error

Theorem 21.4. *If no underflow or overflows are encountered then Algorithm* 21.2 *runs to completion and the computed solution* $\widehat{a}$ *satisfies*

$$|a-\widehat{a}| \le c(n,u)\,|U_0|\dots|U_{n-1}||L_{n-1}|\dots|L_0||f|, \qquad (21.18)$$

where $c(n, u) := (1 + u)^{7n} - 1 = 7nu + O(u^2)$.

Proof. First, note that Algorithm 21.2 must succeed in the absence of underflow and overflow, because division by zero cannot occur.

The analysis of the computation of the $c^{(k)}$ vectors is exactly the same as that for the nonconfluent divided differences in §5.3 (see (5.9) and (5.10)). However, we obtain a slightly cleaner error bound by dropping the γ_k notation and instead writing

$$\widehat{c}^{(k+1)} = (L_k + \Delta L_k)\widehat{c}^{(k)}, \qquad |DL_k| \le [(1+u)^3 - 1]|L_k|. \qquad (21.19)$$

Turning to the equations (21.16), we can regard the multiplication $a^{(k)} = U_k a^{(k+l)}$ as comprising a sequence of three-term inner products. Analysing these in standard fashion we arrive at the equation

$$\widehat{a}^{(k)} = (U_k + \Delta U_k)\widehat{a}^{(k+1)}, \qquad |DU_k| \le [(1+u)^4 - 1]|U_k|, \qquad (21.20)$$

where we have taken into account the rounding errors in forming $u^{(k)}_{i,i+1}$ $\beta_j - \alpha_k$ and $u^{(k)}_{i,i+2} = \gamma_{j+1}/\theta_{j+1}$ $(i = k + j)$.

Since $\widehat{c}^{(0)} = f$, and $\widehat{a} = \widehat{a}^{(0)}$, (21.19) and (21.20) imply that

$$\widehat{a} = (U_0+\Delta U_0)\dots(U_{n-1}+\Delta U_{n-1})(L_{n-1}+\Delta L_{n-1})\dots(L_0+\Delta L_0)f. \qquad (21.21)$$

Applying Lemma 3.7 to (21.21) and using (21.17), we obtain the desired bound for the forward error. □

The product $|U_0| \dots |U_{n-1}||L_{n-1}| \dots |L_0|$ in (21.18) is an upper bound for $|U_0 \dots U_{n-1}L_{n-1} \dots L_0| = |P^{-T}|$ and is equal to it when there is no subtractive cancellation in the latter product. To gain insight, suppose the points are distinct and consider the case $n = 3$. We have

$$\begin{aligned} P^{-T} &= U_0U_1U_2L_2L_1L_0 \\ &\equiv \begin{bmatrix} 1 & \beta_0-\alpha_0 & \gamma_1/\theta_1 & 0 \\ & \theta_0^{-1} & \beta_1-\alpha_0 & \gamma_2/\theta_2 \\ & & \theta_1^{-1} & \beta_2-\alpha_0 \\ & & & \theta_2^{-1} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & \beta_0-\alpha_1 & \gamma_1/\theta_1 \\ & & \theta_0^{-1} & \beta_1-\alpha_1 \\ & & & \theta_1^{-1} \end{bmatrix} \end{aligned}$$

$$\times \begin{bmatrix} 1 & 0 & 0 & 0 \\ & 1 & 0 & 0 \\ & & 1 & \beta_0 - \alpha_2 \\ & & & \theta_0^{-1} \end{bmatrix} \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & 0 & 1 & \\ 0 & 0 & \frac{-1}{\alpha_3-\alpha_0} & \frac{1}{\alpha_3-\alpha_0} \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & & & \\ 0 & 1 & & \\ 0 & \frac{-1}{\alpha_2-\alpha_0} & \frac{1}{\alpha_2-\alpha_0} & \\ 0 & 0 & \frac{-1}{\alpha_3-\alpha_1} & \frac{1}{\alpha_3-\alpha_1} \end{bmatrix}$$

$$\times \begin{bmatrix} 1 & & & \\ \frac{-1}{\alpha_1-\alpha_0} & \frac{1}{\alpha_1-\alpha_0} & & \\ 0 & \frac{-1}{\alpha_2-\alpha_1} & \frac{1}{\alpha_2-\alpha_1} & \\ 0 & 0 & \frac{-1}{\alpha_3-\alpha_2} & \frac{1}{\alpha_3-\alpha_2} \end{bmatrix}. \qquad (21.22)$$

There is no subtractive cancellation in this product as long as each matrix has the alternating (checkerboard) sign pattern defined, for $A = (a_{ij})$, by $(-1)^{i+j}a_{ij} \geq 0$. This sign pattern holds for the matrices L_i if the points α_i are arranged in increasing order. The matrices U_i have the required sign pattern provided that (in general)

$$q_i > 0, \quad \gamma_i > 0 \quad \text{for all } i, \quad \text{and} \quad \beta_i - \alpha_k < 0 \text{ for all } i + k < n - 1.$$

In view of Table 21.2 we have the following result.

Corollary 21.5. *If* $0 \leq \alpha_0 < \alpha_1 < \cdots < \alpha_n$ *then for the monomials, or the Chebyshev, Legendre, or Hermite polynomials,*

$$|a - \widehat{a}| \leq c(n,u)\,|P^{-T}||f|. \qquad \square$$

Corollary 21.5 explains the high accuracy observed by Björck and Pereyra. Note that if

$$|P^{-T}||f| \leq t_n|P^{-T}f| = t_n|a|$$

then, under the conditions of the corollary, $|\widehat{a} - a| \leq c(n,u)t_n|a|$, which shows that the componentwise relative error is bounded by $c(n, u)t_n$. For the problem of Björck and Pereyra it can be shown that $t_n \approx n^4/24$. Another factor contributing to the high accuracy in this problem is that many of the subtractions $\alpha_j - \alpha_{j-k-1} = 1/(j + 3) - 1/(j - k + 2)$ are performed exactly, in view of Theorem 2.5.

Note that under the conditions of the corollary P-T has the alternating sign pattern, since each of its factors does. Thus if $(-1)^i f_i \geq 0$ then $t_n = 1$, and the corollary implies that $\widehat{a}$ is accurate essentially to full machine precision, *independent of the condition number* $\kappa_\infty(P)$. In particular, taking f to be a column of the identity matrix shows that we can compute the inverse of P to high relative accuracy, independent of its condition number.

21.3.2. Residual

Next we look at the residual, $r = f - P^T\widehat{a}$. Rewriting (21.21),

$$f = (L_0+\Delta L_0)^{-1}\dots(L_{n-1}+\Delta L_{n-1})^{-1}(U_{n-1}+\Delta U_{n-1})^{-1}\dots(U_0+\Delta U_0)^{-1}\widehat{a}. \tag{21.23}$$

From the proof of Theorem 21.4 and the relation (5.9) it is easy to show that

$$(L_k + DL_k)^{-1} = L_k^{-1} + E_k, \qquad |E_k| \le [(1-u)^{-3} - 1]|L_k^{-1}|$$

Strictly, an analogous bound for $(U_k + DU_k)^{-1}$ does not hold, since DU_k cannot be expressed in the form of a diagonal matrix times U_k. However, it seems reasonable to make a simplifying assumption that such a bound is valid, say

$$(U_k + DU_k)^{-1} = U_k^{-1} + F_k, \qquad |F_k| \le [(1-u)^{-4} - 1]|U_k^{-1}|. \tag{21.24}$$

Then, writing (21.23) as

$$f = (L_0^{-1} + E_0)\dots(L_{n-1}^{-1} + E_{n-1})(U_{n-1}^{-1} + F_{n-1})\dots(U_0^{-1} + F_0)\widehat{a}$$

and invoking Lemma 3.7, we obtain the following result.

Theorem 21.6. *Under the assumption* (21.24), *the residual of the computed solution* $\widehat{a}$ *from Algorithm* 21.2 *is bounded by*

$$|f - P^T\widehat{a}| \le d(n,u)\,|L_0^{-1}|\dots|L_{n-1}^{-1}||U_{n-1}^{-1}|\dots|U_0^{-1}||\widehat{a}|, \tag{21.25}$$

where $d(n, u) := (1-u)^{-7n} - 1 = 7nu + O(u^2)$. $\square$

For the monomials, with distinct, nonnegative points arranged in increasing order, the matrices L_i and U_i are bidiagonal with the alternating sign property,, as we saw above. Thus $L_{i-}^{1} \ge 0$ and $U_i^{-1} \ge 0$, and since $P^T = L_0^{-1}\dots L_{n-1}^{-1}U_{n-1}^{-1}\dots U_0^{-1}$, we obtain from (21.25) the following pleasing result, which guarantees a tiny componentwise relative backward error.

Corollary 21.7. *Let* $0 \le \alpha_0 < \alpha_1 < \cdots < \alpha_n$, *and consider Algorithm* 21.2 *for the monomials. Under the assumption* (21.24), *the computed solution* $\widehat{a}$ *satisjies*

$$|f - P^T\widehat{a}| \le d(n,u)\,|P^T||\widehat{a}|. \qquad \square$$

Table 21.3. *Results for dual Chebyshev-Vandermonde-like system.*

n	10	20	30	40
$\dfrac{\Vert a-\widehat{a}\Vert_\infty}{\Vert a\Vert_\infty}$	$2.5 \times 10^{-1}2$	6.3×10^{-7}	4.7×10^{-2}	1.8×10^{3}
$\dfrac{\Vert f-P^T\widehat{a}\Vert_\infty}{\Vert P\Vert_\infty \Vert \widehat{a}\Vert_\infty + \Vert f\Vert_\infty}$	6.0×10^{-13}	$1.1 \times 10\text{-}^{7}$	$5.3 \times 10\text{-}^{3}$	8.3×10^{-2}

21.3.3. Dealing with Instability

The potential instability of Algorithm 21.2 is illustrated by the following example. Take the Chebyshev polynomials T_i with the points $\alpha_i = \cos(i\pi/n)$ (the extrema of T_n), and define the right-hand side by $f_i = (-1)^i$. The exact solution to $P^Ta = f$ is the last column of the identity matrix. Relative errors and residuals are shown in Table 21.3 ($u = 10^{-16}$). Even though $_{k2}(P) \le 2$ for all n (see Problem 21.7), the forward errors and relative residuals are large and grow with *n*. The reason for the instability is that there are large intermediate numbers in the algorithm (at the start of stage II for n = 40, $\Vert c\Vert_\infty$ is of order 10^{15}); hence severe cancellation is necessary to produce the final answer of order 1. Looked at another way, the factorization of P^T used by the algorithm is unstable because it is very sensitive to perturbations in the factors.

How can we overcome this instability? There are two possibilites: prevention and cure. The only means at our disposal for *preventing* instability is to reorder the points α_i. The ordering is arbitrary subject to condition (21.5) being satisfied. Recall that the algorithms construct an *LU* factorization of P^T in factored form, and note that permuting the rows of P^T is equivalent to reordering the points α_i. A reasonable approach is therefore to take whatever ordering of the points would be produced by Gaussian elimination with partial pivoting (GEPP) applied to P^T. Since the diagonal elements of *U* in $P^T = LU$ have the form

$$u_{ii} = h_i \prod_{j=0}^{i-1} (\alpha_i - \alpha_j), \quad i = 0{:}n,$$

where ha depends only on the q_i, and since partial pivoting maximizes the pivot at each stage, this ordering of the α_i can be computed at the start of the algorithm in n^2 flops (see Problem 21.8). This ordering is essentially the Leja ordering (5.13) (the difference is that partial pivoting leaves α_0 unchanged).

To attempt to *cure* observed instability we can use iterative refinement in fixed precision. Ordinarily, residual computation for linear equations is trivial, but in this context the coefficient matrix is not given explicitly and computing the residual turns out to be conceptually almost as difficult, and computationally as expensive, as solving the linear system!

To compute the residual for the dual system we need a means for evaluating $y(t)$ in (21.7) and its first $k \le n$ derivatives, where $k = \max_i(i - r(i))$ is the order of confluence. Since the polynomials p_j satisfy a three-term recurrence relation we can use an extension of the Clenshaw recurrence formula (which evaluates y but not its derivatives). The following algorithm implements the appropriate recurrences, which are given by Smith [927, 1965] (see Problem 21.9).

Algorithm 21.8 (extended Clenshaw recurrence). This algorithm computes the $k + 1$ values $y_j = y^{(j)}(x)$, $0 \le j \le k$, where y is given by (21.7) and $k \le n$. It uses a work vector z of order k.

```
Set $y(0\colon k) = z(0\colon k) = 0$
$y_0 = a_n$
for $j = n - 1\colon -1\colon 0$
    $temp = y_0$
    $y_0 = q_j(x - b_j)y_0 - \gamma_{j+1}z_0 + a_j$
    $z_0 = temp$
    for $i = 1\colon \min(k, n - j)$
        $temp = yi$
        $y_i = q_j((x - b_j)y_i + z_{i-1}) - \gamma_{j+1}z_i$
        $z_i = temp$
    end
end
$m = 1$
for $i = 2\colon k$
    $m = m * i$
    $y_i = m * y_i$
end
```

Computing the residual using Algorithm 21.8 costs between $3n^2$ flops (for full confluence) and $6n^2$ flops (for the nonconfluent case); recall that Algorithm 21.2 costs at most $9n^2/2$ flops!

The use of iterative refinement can be justified with the aid of Theorem 11.3. For (confluent) Vandermonde matrices, the residuals are formed using Homer's rule and (11 .7) holds in view of (5.3) and (5.7). Hence for standard Vandermonde matrices, Theorem 11.3 leads to an asymptotic componentwise backward stability result. A complete error analysis of Algorithm 21.8 is not available for (confluent) Vandermonde-like matrices, but it is easy to see that (11.7) will not always hold. Nevertheless it is clear that a *normwise* bound can be obtained (see Oliver [805, 1977] for the special case of the Chebyshev polynomials) and hence an asymptotic normwise stability result can be deduced from Theorem 11.3. Thus there is theoretical backing for the use of iterative refinement with Algorithm 21.8.

Numerical experiments using Algorithm 21.8 in conjunction with the partial pivoting reordering and fixed precision iterative refinement show that both techniques are effective means for stabilizing the algorithms, but that iterative refinement is likely to fail once the instability y is sufficiently severe. Because of its lower cost, the reordering approach is preferable.

Two heuristics are worth noting. Consider a (confluent) Vandermonde-like system $Px = b$ *Heuristic* 1: it is systems with a large normed solution $(\|x\| \approx \|P^{-1}\|\,\|b\|)$ that are solved to high accuracy by the fast algorithms. To produce a large solution the algorithms must sustain little cancellation, and the error analysis shows that cancellation is the main cause of instability. *Heuristic* 2: GEPP is unable to solve accurately Vandermonde systems with a very large normed solution $(\|x\| \gg u^{-1}\|b\|/\|P\|)$. The pivots for GEPP will tend to satisfy $|u_{ii}| \gtrsim u\|P\|$, so that the computed solution will tend to satisfy $\|\widehat{x}\| \le u^{-1}\|b\|/\|P\|$. A consequence of these two heuristics is that for Vandermonde(-like) systems with a very large-normed solution the fast algorithms will be much more accurate (but no more backward stable) than GEPP. However, we should be suspicious of any framework in which such systems arise; although the solution vector may be obtained accurately (barring overflow), subsequent computations with numbers of such a wide dynamic range will probably themselves be unstable.

21.4. Notes and References

The formulae (21. 1) and (21.2), and inversion methods based on these formulae, have been discovered independently by many authors. Traub [1013, 1966, §14] gives a short historical survey, his earliest reference being a 1932 book by Kowalewski. There does not appear to be any published error analysis for Algorithm 21.1 (see Problem 21.3). There is little justification for using the output of the algorithm to solve the primal or dual linear system, as is done in [842, 1992, §2.8]; Algorithms 21.2 and 21.3 are more efficient and almost certainly at least as stable. Calvetti and Reichel [179, 1993] generalize Algorithm 21.1 to Vandermonde-like matrices, but they do not present any error analysis. Gohberg and Olshevsky [456, 1994] give another $O(n^2)$ flops algorithm for inverting a Chebyshev–Vandermonde matrix.

The standard condition number $\kappa(V)$ is not an appropriate measure of sensitivity when only the points α_i are perturbed, because it does not reflect the special structure of the perturbations. Appropriate condition numbers were first derived by Higham [533, 1987] and are comprehensively investigated by Bartels and D. J. Higham [76, 1992]; see Problem 21.10.

Methods for solving the dual and primal Vandermonde systems have an interesting history. The earliest algorithm was derived by Lyness and Moler [718, 1966] via Neville interpolation; it solves the dual system in $O(n^3)$ flops. The first $O(n^2)$ algorithm was obtained by Ballester and Pereyra [52, 1967]; it computes the LU factors of the Vandermonde matrix and requires $O(n^2)$ elements of storage. Björck and Pereyra [121, 1970] derived the specialization of Algorithms 21.2 and 21.3 to nonconfluent Vandermonde matrices; these algorithms require no storage other than that for the problem data. The algorithms of Björck and Pereyra were generalized by Björck and Elfving to confluent systems [117, 1973], and by Higham to Vandermonde-like systems [536, 1988] and confluent Vandermonde-like systems [547, 1990]. The error analysis in this chapter is taken from [547, 1990]. Tang and Golub [992, 1981] give a block algorithm that requires only real arithmetic to solve a Vandermonde system in which all the points appear in complex conjugate pairs.

Other $O(n^2)$ algorithms for solving Chebyshev-Vandermonde systems are given by Reichel and Opfer [865, 1991] and Calvetti and Reichel [178, 1992]. The former algorithms are progressive, in that they allow the solution to be updated when a new point α_i is added; they generalize progressive algorithms of Björck and Pereyra [121, 1970]. Boros, Kailath, and Olshevsky [135, 1994] use the concept of displacement structure to derive further $O(n^2)$ algorithms for solving Vandermonde and Chebyshev–Vandermonde systems. No error analysis is given in [135, 1994], [178, 1992], or [865, 1991].

The $O(n^2)$ complexity of the algorithms mentioned above for solving Vandermonde-like systems is not optimal. Lu [714, 1994], [715, 1995], [716, 1996] derives $O(n \log n \log p)$ flops algorithms, where p is the number of distinct points. The numerical stability and practical efficiency of the algorithms remain to be determined. Bini and Pan [98, 1994] give an $O(n \log^2 n)$ algorithm for solving a dual Vandermonde system that involves solving related Toeplitz and Hankel systems.

Since Vandermonde systems can be solved in less than $O(n^3)$ flops it is natural to ask whether the $O(mn^2)$ complexity of QR factorization of an m x n matrix can be bettered for a Vandermonde matrix. QR factorization algorithms with $O(mn)$ flop counts have been derived by Demeure [277, 1989], [278, 1990], and for Vandermonde-like matrices where the polynomials satisfy a three-term recurrence by Reichel [864, 1991]. No error analysis has been published for these algorithms. Demeure's algorithms are likely to be unstable, because they form the normal equations matrix V^TV.

Problems

21.1. Derive a modified version of Algorithm 21.1 in which the scale factor

$s = q(\alpha_i)$ is computed directly as

$$\prod_{\substack{k=0\\k\neq i}}^{n}(\alpha_i-\alpha_k).$$

What is the flop count for this version?

21.2. (Calvetti and Reichel [179, 1993]) Generalize Algorithm 21.1 to the inversion of a Vandermonde-like matrix for polynomials that satisfy a three-term recurrence relation.

21.3. Investigate the stability of Algorithm 21.1 and the modified version of Problem 21.1. (a) Evaluate the left and right residuals of the computed inverses; compare the results with those for GEPP. (b) Show that Algorithm 21.1 always performs subtractions of like-signed numbers and explain the implications for stability. (Does Algorithm 21.1 share the high accuracy properties discussed at the end of 521.3.1?) (c) (RESEARCH PROBLEM) Derive and explore forward error bounds and residual bounds for both algorithms. Extend the analysis to the algorithms of Calvetti and Reichel [179, 1993].

21.4. By summing (21.1) for $i = 0{:}n$, show that $\sum_{i=0}^{n} w_{ij} = \delta_{j0}$. What does this imply about the sign pattern of V^{-1} ? What is the sum of all the elements of V^{-1}?

21.5. Let $T = T(\alpha_0, \alpha_1, \ldots, \alpha_n) = (T_i(\alpha_j))_{i,j=0}^{n}$ be a Chebyshev–Vandermonde matrix (T_j is the Chebyshev polynomial of degree j), with $T^{-1} = (u_{ij})$. Analogously to (21.1) we have $\sum_{j=0}^{n} u_{ij}T_j(x) = l_i(x)$. Hence $\sum_{j=0}^{n} w_{ij}x^j = \sum_{j=0}^{n} u_{ij}T_j(x)$, where $V^{l} = V(\alpha_0, \alpha_1, \ldots, \alpha_n)^{-1} = (w_{ij})$. Show that

$$\sum_{j=0}^{n}|u_{ij}| \le \sum_{j=0}^{n}|w_{ij}|,$$

and hence that

$$\|T^{-1}\|_\infty \le \|V^{-1}\|_\infty.$$

(Hint: show that $T = LV$ for a lower triangular matrix L and use the fact that $x^n = \frac{1}{2^{n+1}}\sum_{k=0}^{\lceil n/2\rceil}\binom{n}{k}T_{n-2k}(x)$.)

21.6. Show that for a nonconfluent Vandermonde-like matrix $P = (P_i(\alpha_j))_{i,j=0}^{n}$, where the p_i satisfy (21.6),

$$\det(P) = \prod_{i=0}^{n-1}\theta_i^{n-i}\prod_{i>j}(\alpha_i-\alpha_j).$$

(Hint: study the structure of (21.22).)

21.7. Show that for the Chebyshev-Vandermonde-like matrix $T = T(\alpha_0, a_1, \ldots, a_n)$,

1. $\kappa_2(T) = \sqrt{2}$, for $\alpha_i = \cos((i + \frac{1}{2})\pi/(n + 1))$ (zeros of T_{n+1}).
2. $\kappa_2(T) \leq 2$, for $\alpha_i = \cos(i\pi/n)$ (extrema of T_n).

(Hint: use the discrete orthogonality properties of the Chebyshev polynomials; see, e.g., Hamming [501, 1973, pp.472–473].)

21.8. Derive an $O(n^2)$ flops algorithm that reorders the distinct points $\alpha_0, \alpha_1, \ldots, \alpha_n \in \mathbb{C}$ according to the same permutation that would be produced by GEPP applied to $P^T(\alpha_0, \alpha_1, \ldots, \alpha_n)$. (Cf. Problem 5.4.) Can your algorithm ever produce the increasing ordering?

21.9. Derive Algorithm 21.8 by differentiating repeatedly the original Clenshaw recurrence (which is Algorithm 21.8 with $k = 0$) and resealing so as to consign factorial terms to a cleanup loop at the end. Derive an algorithm for computing the residual for the primal system in a similar way, using recurrences obtained by differentiating (21.6).

21.10. (Higham [533, 1987]) A structured condition number for the primal Vandermonde system $Vx = b$, where $V = V(\alpha_0, \alpha_1, \ldots, \alpha_n)$, can be defined by

$$\begin{aligned}\operatorname{cond}(V) := \lim_{\epsilon \to 0} \sup\Big\{ \frac{\|\Delta x\|_\infty}{\epsilon \|x\|_\infty} : V(\alpha_0 + \Delta\alpha_0, \ldots, \alpha_n + \Delta\alpha_n)(x + \Delta x) = b, \\ |\Delta\alpha_i| \leq \epsilon|\alpha_i|, \quad i = 0{:}\, n. \Big\}.\end{aligned}$$

Show that

$$\operatorname{cond}(V) = \frac{\|\, |V^{-1} \operatorname{diag}(0, 1, 2, \ldots, n) V| |x| \,\|_\infty}{\|x\|_\infty},$$

and derive a corresponding condition number for the dual system $V^T a = f$.

Previous Home Next

Chapter 22

Fast Matrix Multiplication

A simple but extremely valuable bit of equipment in matrix multiplication
consists of two plain cards,
with a re-entrant right angle cut out of one or both of them
if symmetric matrices are to be multiplied.
In getting the element of the ith row and jth column of the product,
the ith row of the first factor and the jth column of the second
should be marked by a card beside, above, or below it.

— HAROLD HOTELLING, *Some New Methods in Matrix Calculation (1943)*

It was found that multiplication of matrices using punched card storage
could be a highly efficient process on the Pilot ACE,
due to the relative speeds of the Hollerith card reader used for input
(one number per 16 ins.) and the automatic multiplier (2 ins.).
While a few rows of one matrix were held in the machine
the matrix to be multiplied by it was passed through the card reader.
The actual computing and selection of numbers from store
occupied most of the time between the passage of
successive rows of the cards through the reader,
so that the overall time was but little longer
than it would have been if the machine
had been able to accommodate both matrices.

— MICHAEL WOODGER, *The History and Present Use of*
Digital Computers at the National Physical Laboratory (1958)

22.1. Methods

A fast matrix multiplication method forms the product of two n x n matrices in $O(n^\omega)$ arithmetic operations, where $\omega < 3$. Such a method is more efficient asymptotically than direct use of the definition

$$(AB)_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \tag{22.1}$$

which requires $O(n^3)$ operations. For over a century after the development of matrix algebra in the 1850s by Cayley, Sylvester and others, this definition provided the only known method for multiplying matrices. In 1967, however, to the surprise of many, Winograd found a way to exchange half the multiplications for additions in the basic formula [1105, 1968]. The method rests on the identity, for vectors of even dimension n,

$$x^T y = \sum_{i=1}^{n/2} (x_{2i-1} + y_{2i})(x_{2i} + y_{2i-1}) - \sum_{i=1}^{n/2} x_{2i-1} x_{2i} - \sum_{i=1}^{n/2} y_{2i-1} y_{2i}. \tag{22.2}$$

When this identity is applied to a matrix product *AB,* with *x* a row of *A* and *y* a column of *B,* the second and third summations are found to be common to the other inner products involving that row or column, so they can be computed once and reused. Winograd's paper generated immediate practical interest because on the computers of the 1960s floating point multiplication was typically two or three times slower than floating point addition. (On todays' machines these two operations are usually similar in cost).

Shortly after Winograd's discovery, Strassen astounded the computer science community by finding an $O(n^{\log_2 7})$ operations method for matrix multiplication ($\log_2 7 \approx 2.807$). A variant of this technique can be used to compute A^{-1} (see Problem 22.8) and thereby to solve $Ax = b$, both in $O(n^{\log_2 7})$ operations. Hence the title of Strassen's 1969 paper [962, 1969], which refers to the question of whether Gaussian elimination is asymptotically optimal for solving linear systems.

Strassen's method is based on a circuitous way to form the product of a pair of 2 x 2 matrices in 7 multiplications and 18 additions, instead of the usual 8 multiplications and 4 additions. As a means of multiplying 2 x 2 matrices the formulae have nothing to recommend them, but they are valid more generally for block 2 x 2 matrices. Let *A* and *B* be matrices of dimensions m x n and *n* x *p* respectively, where all the dimensions are even, and partition each of *A,* *B,* and *C* = *AB* into four equally sized blocks:

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}. \tag{22.3}$$

Strassen's formulae are

$$\begin{aligned}
P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
P_2 &= (A_{21} + A_{22})B_{11}, \\
P_3 &= A_{11}(B_{12} - B_{22}), \\
P_4 &= A_{22}(B_{21} - B_{11}), \\
P_5 &= (A_{11} + A_{12})B_{22}, \\
P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}) \\
P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
C_{11} &= P_1 + P_4 - P_5 + P_7, \\
C_{12} &= P_3 + P_5, \\
C_{21} &= P_2 + P_4, \\
C_{22} &= P_1 + P_3 - P_2 + P_6.
\end{aligned} \tag{22.4}$$

Counting the additions (A) and multiplications (M) we find that while conventional multiplication requires

$$mnpM + m(n-1)pA,$$

Strassen's algorithm, using conventional multiplication at the block level, requires

$$\frac{7}{8}mnpM + \left(\frac{7}{8}m(n-2)p + \frac{5}{4}mn + \frac{5}{4}np + \frac{8}{4}mp\right)A.$$

Thus, if m, n, and p are large, Strassen's algorithm reduces the arithmetic by a factor of about 7/8. The same idea can be used recursively on the multiplications associated with the Pa. In practice, recursion is only performed down to the "crossover" level at which any savings in floating point operations are outweighed by the overheads of a computer implementation.

To state a complete operation count, we suppose that $m = n = p = 2^k$ and that recursion is terminated when the matrices are of dimension no $= 2^r$, at which point conventional multiplication is used. The number of multiplications and additions can be shown to be

$$M(k) = 7^{k-r}8^r, \qquad A(k) = 4^r(2^r+5)7^{k-r} - 6\cdot 4^k. \tag{22.5}$$

The sum $M(k) + A(k)$ is minimized over all integers r by $r = 3$; interestingly, this value is independent of k. The total operation count for the "optimal" no $= 8$ is less than

$$4\cdot 7^k = 4\cdot 2^{\log_2 7^k} = 4(2^k)^{\log_2 7} = 4n^{\log_2 7} = 4n^{2.807\ldots}.$$

Hence, in addition to having a lower exponent, Strassen's method has a reasonable constant.

Winograd found a variant of Strassen's formulae that requires the same number of multiplications but only 15 additions (instead of 18). This variant therefore has slightly smaller constants in the operation count for n x n matrices. For the product (22.3) the formulae are

$$\begin{array}{lll}
S_1 = A_{21} + A_{22}, & M_1 = S_2 S_6, & T_1 = M_1 + M_2, \\
S_2 = S_1 - A_{11}, & M_2 = A_{11}B_{11}, & T_2 = T_1 + M4, \\
S_3 = A_{11} - A_{21}, & M_3 = A_{12}B_{21}, & \\
S_4 = A_{12} - S_2, & M_4 = S_3 S_7, & \\
S_5 = B_{12} - B_{11}, & M_5 = S_1 S_5, & C_{11} = M_2 + M_3, \\
S_6 = B_{22} - S_5, & M_6 = S_4 B_{22}, & C_{12} = T_I + M_5 + M_6, \\
S_7 = B_{22} - B_{12}, & M_7 = A_{22} S_8, & C_{21} = T_2 - M_7, \\
S_8 = S_6 - B_{21}, & & C_{22} = T_2 + M_5.
\end{array} \qquad (22.6)$$

Until the late 1980s there was a widespread view that Strassen's method was of theoretical interest only, because of its supposed large overheads for dimensions of practical interest (see, for example, [909, 1988]), and this view is still expressed by some [842, 1992]. However, in 1970 Brent implemented Strassen's algorithm in Algol-W on an IBM 360/67 and concluded that in this environment, and with just one level of recursion, the method runs faster than the conventional method for $n \geq 110$ [142, 1970]. In 1988, Bailey compared his Fortran implementation of Strassen's algorithm for the Cray-2 with the Cray library routine for matrix multiplication and observed speedup factors ranging from 1.45 for n = 128 to 2.01 for n = 2048 (although 35% of these speedups were due to Cray-specific techniques) [43, 1988]. These empirical results, together with more recent experience of various researchers, show that Strassen's algorithm *is* of practical interest, even for n in the hundreds. Indeed, Fortran codes for (Winograd's variant of) Straasen's method have been supplied with IBM's ESSL library [595, 1988] and Cray's UNICOS library [602, 1989] since the late 1980s.

Strassen's paper raised the question "what is the minimum exponent ω such that multiplication of n x n matrices can be done in $O(n^\omega)$ operations?" Clearly, $\omega \geq 2$, since each element of each matrix must partake in at least one operation. It was 10 years before the exponent was reduced below Strassen's $\log_2 7$. A flurry of publications, beginning in 1978 with Pan and his exponent 2.795 [815, 1978], resulted in reduction of the exponent to the current record 2.376, obtained by Coppersmith and Winograd in 1987 [245, 1987]. Figure 22.1 plots exponent versus time of publication (not all publications are represented in the graph); in principle, the graph should extend back to 1850!

Some of the fast multiplication methods are based on a generalization of Strassen's idea to bilinear forms. Let $A, B \in \mathbb{R}^{n \times n}$. A bilinear noncommuta-

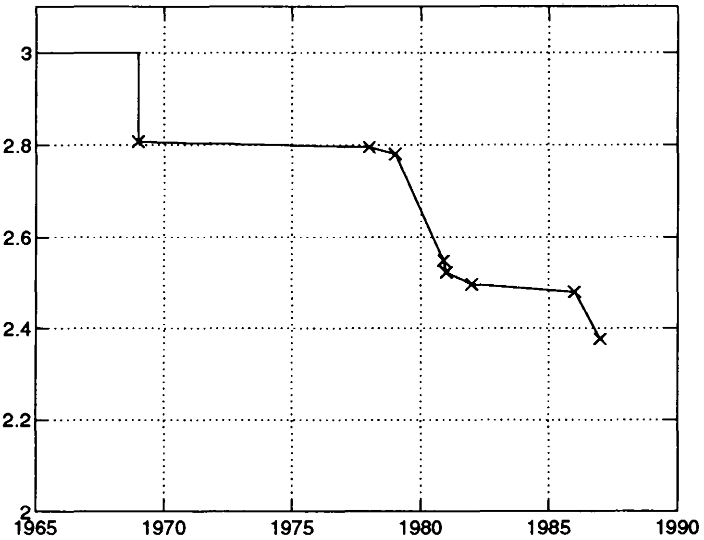


Figure 22.1. *Exponent versus time for matrix multiplication.*

tive algorithm over $\mathbb{R}$ for multiplying A and B with t "nonscalar multiplications" forms the product $C = AB$ according to

$$c_{ij} = \sum_{k=1} w_{h(i-1)+j,k} P_k, \tag{22.7a}$$

$$P_k = \left(\sum_{i,j=1}^{h} u_{ij}^{(k)} a_{ij} \right) \left(\sum_{i,j=1}^{h} v_{ij}^{(k)} b_{ij} \right), \quad k = 1\colon t, \tag{22.7b}$$

where the elements of the matrices W, $U^{(k)}$, and $V^{(k)}$ are constants. This algorithm can be used to multiply n x n matrices A and B, where $n = h^k$, as follows: partition A, B, and C into h^2 blocks A_{ij}, B_{ij}, and C_{ij} of size h^{k-1}, then compute $C = AB$ by the bilinear algorithm, with the scalars a_{ij}, b_{ij}, and c_{ij} replaced by the corresponding matrix blocks. (The algorithm is applicable to matrices since, by assumption, the underlying formulae do not depend on commutativity.) To form the t products P_k of (n/h) x (n/h) matrices, partition them into h^2 blocks of dimension n/h^2 and apply the algorithm recursively. The total number of scalar multiplications required for the multiplication is $t^k = n^{\alpha}$, where $\alpha = \log_h t$.

Strassen's method has $h = 2$ and $t = 7$. For 3 x 3 multiplication ($h = 3$), the smallest t obtained so far is 23 [683, 1976]; since $\log_3 23 \approx 2.854 > \log_2 7$, this does not yield any improvement over Strassen's method. The method described in Pan's 1978 paper has $h = 70$ and $t = 143{,}640$, which yields $\alpha = \log_{70} 143{,}640 = 2.795\ldots$.

In the methods that achieve exponents lower than 2.775, various intricate techniques are used. Laderman, Pan, and Sha [684, 1992] explain that for these methods "very large overhead constants are hidden in the 'O' notation", and that the methods "improve on Strassen's (and even the classical) algorithm only for immense numbers N".

A further method that is appropriate to discuss in the context of fast multiplication methods, even though it does not reduce the exponent, is a method for efficient multiplication of complex matrices. The clever formula

$$(a + ib)(c + id) = ac - bd + i[(a + b)(c + d) - ac - bd] \tag{22.8}$$

computes the product of two complex numbers using three real multiplications instead of the usual four. Since the formula does not rely on commutativity it extends to matrices. Let $A = A_1 + iA_2$ and $B = B_1 + iB_2$, where $A_j, B_j \in \mathbb{R}^{n\times n}$, and define $C = C_1 + iC_2 = AB$. Then C can be formed using three real matrix multiplications as

$$\begin{aligned} T_1 &= A_1B_1, \qquad T_2 = A_2B_2, \\ C_1 &= T_1 - T_2, \\ C_2 &= (A_1 + A_2)(B_1 + B_2) - T_1 - T_2, \end{aligned} \tag{22.9}$$

which we will refer to as the "3M method". This computation involves $3n^3$ scalar multiplications and $3n^3 + 2n^2$ scalar additions. Straightforward evaluation of the conventional formula $C = A_1B_1 - A_2B_2 + i(A_1B_2 + A_2B_1)$ requires $4n^3$ multiplications and $4n^3 - 2n^2$ additions. Thus, the 3M method requires strictly fewer arithmetic operations than the conventional means of multiplying complex matrices for $n \geq 3$, and it achieves a saving of about 25% for $n \geq 30$ (say). Similar savings occur in the important special case where A or B is triangular. This kind of clear-cut computational saving is rare in matrix computations!

IBM's ESSL library and Cray's UNICOS library both contain routines for complex matrix multiplication that apply the 3M method and use Strassen's method to evaluate the resulting three real matrix products.

22.2. Error Analysis

To be of practical use, a fast matrix multiplication method needs to be faster than conventional multiplication for reasonable dimensions without sacrificing numerical stability. The stability properties of a fast matrix multiplication method are much harder to predict than its practical efficiency, and need careful investigation.

The forward error bound (3.12) for conventional computation of $C = AB$, where A, $B \in \mathbb{R}^{n\times n}$, can be written

$$|C-\widehat{C}| \le nu|A||B| + O(u^2). \tag{22.10}$$

Miller [756, 1975] shows that any polynomial algorithm for multiplying n x n matrices that satisfies a bound of the form (22.10) (perhaps with a different constant) must perform at least n^3 multiplications. (A polynomial algorithm is essentially one that uses only scalar addition, subtraction, and multiplication.) Hence Strassen's method, and all other polynomial algorithms with an exponent less than 3, cannot satisfy (22.10). Miller also states, without proof, that any polynomial algorithm in which the multiplications are all of the form $\big(\sum_{i,j} u_{ij}^{(k)} a_{ij}\big)\big(\sum_{i,j} v_{ij}^{(k)} b_{ij}\big)$ must satisfy a bound of the form

$$|C-\widehat{C}| \le f_n u\|A\|\,\|B\| + O(u^2). \tag{22.11}$$

It follows that any algorithm based on recursive application of a bilinear noncommutative algorithm satisfies (22.11); however, the all-important constant f_n is not specified. These general results are useful because they show us what types of results we can and cannot prove and thereby help to focus our efforts.

In the subsections below we analyse particular methods.

Throughout the rest of this chapter an unsubscripted matrix norm denotes

$$\|A\| := \max_{i,j} |a_{ij}|.$$

As noted in §6.2, this is not a consistent matrix norm, but we do have the bound $\|AB\| \le n\|A\|\,\|B\|$ for n x n matrices.

22.2.1. Winograd's Method

Winograd's method does not satisfy the conditions required for the bound (22.11), since it involves multiplications with operands of the form $a_{ij} + b_{rs}$. However, it is straightforward to derive an error bound.

Theorem 22.1 (Brent). *Let x, $y \in \mathbb{R}^{n\times n}$, where n is even. The inner product computed by Winograd's method satisfies*

$$|x^Ty - fl(x^Ty)| \le n\gamma_{n/2+4}(\|x\|_\infty + \|y\|_\infty)^2. \tag{22.12}$$

Proof. A straightforward adaptation of the inner product error analysis in §3.1 produces the following analogue of (3.3):

$$fl(x^Ty) = \sum_{i=1}^{n/2}(x_{2i-1}+y_{2i})(x_{2i}+y_{2i-1})(1+\epsilon_i)$$

$$-\sum^{n/2} x_{2i-1}x_{2i}(1+\alpha_i) - \sum^{n/2} y_{2i-1}y_{2i}(1+\beta_i),$$

where the ϵ_i, α_i, and b_i are all bounded in modulus by $\gamma_{n/2+4}$. Hence

$$\begin{aligned} |x^Ty - fl(x^Ty)| &\le \gamma_{n/2+4} \sum_{i=1}^{n/2} \big(|x_{2i-1} + y_{2i}||x_{2i} + y_{2i-1}| \\ &\qquad\qquad + |x_{2i-1}x_{2i}| + |y_{2i-1}y_{2i}|\big) \\ &\le \gamma_{n/2+4}(n/2)\big((\|x\|_\infty + \|y\|_\infty)^2 + \|x\|_\infty^2 + \|y\|_\infty^2\big) \\ &\le n\gamma_{n/2+4}(\|x\|_\infty + \|y\|_\infty)^2. \qquad \square \end{aligned}$$

The analogue of (22.12) for matrix multiplication is $\|AB - fl(AB)\| \le n\gamma_{n/2+4}(\|A\| + \|B\|)^2$.

Conventional evaluation of x^Ty yields the bound (see (3.5))

$$|x^Ty - fl(x^Ty)| \le \gamma_n |x|^T|y| \le n\gamma_n \|x\|_\infty \|y\|_\infty. \tag{22.13}$$

The bound (22. 12) for Winograd's method exceeds the bound (22.13) by a factor approximately $\|x\|_\infty/\|y\|_\infty + \|y\|_\infty/\|x\|_\infty$. Therefore Winograd's method is stable if $\|x\|_\infty$ and $\|y\|_\infty$ have similar magnitude, but potentially unstable if they differ widely in magnitude. The underlying reason for the instability is that Winograd's method relies on cancellation of terms $x_{2i-1}x_{2i}$ and $y_{2i-1}y_{2i}$ that can be much larger ($\|x\|_\infty^2$ and $\|y\|_\infty^2$) than the final answer ($|x^Ty| \le \|x\|_\infty \|y\|_1$); therefore the intermediate rounding errors can swamp the desired inner product.

A simple way to avoid the instability is to scale $x \leftarrow mx$ and $y \leftarrow m^{-1}y$ before applying Winograd's method, where m, which in practice might be a power of the machine base to avoid roundoff, is chosen so that $\|x\|_\infty \approx \|y\|_\infty$. When using Winograd's method for a matrix multiplication AB it suffices to carry out a single scaling $A \leftarrow mA$ and $B \leftarrow m^{-1}B$ such that $\|A\| \approx \|B\|$. If A and B are scaled so that $t^{-1} \le \|A\|/\|B\| \le t$ then

$$\|AB - fl(AB)\| \le 2(\tau + 1)n\gamma_{n/2+4}\|A\|\,\|B\|.$$

22.2.2. Strassen's Method

Until recently there was a widespread belief that Strassen's method is numerically unstable. The next theorem, originally proved by Brent in 1970, shows that this belief is unfounded.

Theorem 22.2 (Brent). *Let A, $B \in \mathbb{R}^{n\times n}$, where $n = 2^k$. Suppose that $C = AB$ is computed by Strassen's method and that $n_0 = 2^r$ is the threshold at which conventional multiplication is used. The computed product $\widehat{C}$ satisfies*

$$\|C-\widehat{C}\| \le \left[\left(\frac{n}{n_0}\right)^{\log_2 12}(n_0^2+5n_0)-5n\right] u\|A\|\,\|B\| + O(u^2). \tag{22.14}$$

Proof. We will use without comment the norm inequality $\|AB\| \le n\|A\|\,\|B\| = 2^k\|A\|\,\|B\|$.

Assume that the computed product $\widehat{C} \approx AB$ from Strassen's method satisfies

$$\widehat{C} = AB + E, \qquad \|E\| \le c_k u\|A\|\,\|B\| + O(u^2), \tag{22.15}$$

where c_k is a constant. In view of (22.10), (22.15) certainly holds for n = no, with $c_r = n_0^2$. Our aim is to verify (22.15) inductively and at the same time to derive a recurrence for the unknown constant c_k.

Consider C_{11} in (22.4), and, in particular, its subterm P_1. Accounting for the errors in matrix addition and invoking (22.15), we obtain

$$\widehat{P}_1 = (A_{11}+A_{22}+\Delta_A)(B_{11}+B_{22}+\Delta_B)+E_1,$$

where

$$\begin{aligned}
|\Delta_A| &\le u|A_{11}+A_{22}|,\\
|\Delta_B| &\le u|B_{11}+B_{22}|,\\
\|E_1\| &\le c_{k-1}u\|A_{11}+A_{22}+\Delta_A\|\,\|B_{11}+B_{22}+\Delta_B\| + O(u^2)\\
&\le 4c_{k-1}u\|A\|\,\|B\| + O(u^2).
\end{aligned}$$

Hence

$$\begin{aligned}
\widehat{P}_1 &= P_1 + F_1,\\
\|F_1\| &\le (8\cdot 2^{k-1} + 4c_{k-1})u\|A\|\,\|B\| + O(u^2).
\end{aligned}$$

Similarly,

$$\widehat{P}_4 = A_{22}(B_{21} - B_{11} + \Delta_B) + E_4,$$

where

$$\begin{aligned}
|\Delta_B| &\le u|B_{21} - B_{11}|,\\
\|E_4\| &\le c_{k-1}u\|A_{22}\|\,\|B_{21} - B_{11} + \Delta_B\| + O(u^2),
\end{aligned}$$

which gives

$$\begin{aligned}
\widehat{P}_4 &= P_4 + F_4,\\
\|F_4\| &\le (2\cdot 2^{k-1} + 2c_{k-1})u\|A\|\,\|B\| + O(u^2).
\end{aligned}$$

Now

$$\widehat{C}_{11} = fl(\widehat{P}_1 + \widehat{P}_4 - \widehat{P}_5 + \widehat{P}_7),$$

where $\widehat{P}_5 =: P_5 + F_5$ and $\widehat{P}_7 =: P_7 + F_7$ satisfy exactly the same error bounds as $\widehat{P}_4$ and $\widehat{P}_1$, respectively. Assuming that these four matrices are added in the order indicated, we have

$$\begin{aligned}\widehat{C}_{11} &= C_{11} + \Delta C_{11},\\ \|\Delta C_{11}\| &\le u(3\|\widehat{P}_1\| + 3\|\widehat{P}_4\| + 2\|\widehat{P}_5\| + \|\widehat{P}_7\|) + \|F_1 + F_4 - F_5 + F_7\| + O(u^2)\\ &\le 26\cdot 2^{k-1}u\|A\|\,\|B\| + 4(5\cdot 2^{k-1} + 3c_{k-1})u\|A\|\,\|B\| + O(u^2)\\ &= (46\cdot 2^{k-1} + 12c_{k-1})u\|A\|\,\|B\| + O(u^2).\end{aligned}$$

Clearly, the same bound holds for the other three $\|\Delta C_{ij}\|$ terms. Thus, overall,

$$\widehat{C} = AB + \Delta C, \qquad \|\Delta C\| \le (46\cdot 2^{k-1} + 12c_{k-1})u\|A\|\,\|B\| + O(u^2).$$

A comparison with (22.15) shows that we need to define the c_k by

$$c_k = 12c_{k-1} + 46\cdot 2^{k-1}, \quad k > r, \quad c_r = 4^r, \tag{22.16}$$

where $c_r =: n_0^2$. Solving this recurrence we obtain

$$\begin{aligned}c_k &= 12^{k-r}4^r + 46\cdot 2^{k-1}\left(\frac{6^{k-r}-1}{5}\right)\\ &= \left(\frac{n}{n_0}\right)^{\log_2 12} n_0^2 + \frac{46}{5}\frac{n}{2}\left(\left(\frac{n}{n_0}\right)^{\log_2 6} - 1\right)\\ &\le \left(\frac{n}{n_0}\right)^{\log_2 12}(n_0^2 + 5n_0) - 5n,\end{aligned}$$

which gives (22.14). $\square$

The forward error bound for Strassen's method is not of the componentwise form (22.10) that holds for conventional multiplication, which we know it cannot be by Miller's result. One unfortunate consequence is that while the scaling $AB \to (AD)(D^{-1}B)$, where D is diagonal, leaves (22.10) unchanged, it can alter (22.14) by an arbitrary amount.

The reason for the scale dependence is that Strassen's method adds together elements of A matrix-wide (and similarly for B); for example, in (22.4) A_{11} is added to A_{22}, A_{12}, and A_{21}. This intermingling of elements is particularly undesirable when A or B has elements of widely differing magnitudes because it allows large errors to contaminate small components of the product. This phenomenon is well illustrated by the example

$$C = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}1 & \epsilon\\ \epsilon & \epsilon^2\end{bmatrix},$$

which is evaluated exactly in floating point arithmetic if we use conventional multiplication. However, Strassen's method computes

$$c_{22} = 2(1+\epsilon^2) + (\epsilon - \epsilon^2) - 1 - (1+\epsilon).$$

Because c_{22} involves subterms of order unity, the error $c_{22} - \widehat{c}_{22}$ will be of order u. Thus the relative error $|c_{22} - \widehat{c}_{22}|/|c_{22}| = O(u/\epsilon^2)$, which is much larger than u if ε is small. This is an example where Strassen's method does not satisfy the bound (22.10). For another example, consider the product $X = P_{32}E$, where P_n is the n x n Pascal matrix (see §26.4) and $e_{ij} \equiv 1/3$. With just one level of recursion in Strassen's method we find in MATLAB that $\max_{ij} |x_{ij} - \widehat{x}_{ij}|/|x_{ij}|$ is of order 10^{-5}, so that, again, some elements of the computed product have high relative error.

It is instructive to compare the bound (22.14) for Strassen's method with the weakened, normwise version of (22.10) for conventional multiplication:

$$\|C - \widehat{C}\| \le n^2 u \|A\| \, \|B\| + O(u^2). \tag{22.17}$$

The bounds (22. 14) and (22. 17) differ only in the constant term. For Strassen's method, the greater the depth of recursion the bigger the constant in (22.14): if we use just one level of recursion ($n_0 = n/2$) then the constant is $3n^2 + 25n$, whereas with full recursion ($n_0 = 1$) the constant is $6n^{\log_2 12} - 5n \approx 6n^{3.585} - 5n$. It is also interesting to note that the bound for Strassen's method (minimal for $n_0 = n$) is not correlated with the operation count (minimal for $n_0 = 8$).

Our conclusion is that Strassen's method has less favorable stability properties than conventional multiplication in two respects: it satisfies a weaker error bound (normwise rather than componentwise) and it has a larger constant in the bound (how much larger depending on no).

Another interesting property of Strassen's method is that it always involves some genuine subtractions (assuming that all additions are of nonzero terms). This is easily deduced from the formulae (22.4). This makes Strassen's method unattractive in applications where all the elements of A and B are nonnegative (for example, in Markov processes). Here, conventional multiplication yields low componentwise relative error because, in (22.10), $|A||B| = |AB| = |C|$, yet comparable accuracy cannot be guaranteed for Strassen's method.

An analogue of Theorem 22.2 holds for Winograd's variant of Strassen's method.

Theorem 22.3. *Let $A, B \in \mathbb{R}^{n\times n}$, where $n = 2^k$. Suppose that $C = AB$ is computed by Winograd's variant* (22.6) *of Strassen's method and that $n_0 = 2^r$ is the threshold at which conventional multiplication is used. The computed product $\widehat{C}$ satisfies*

$$\|C - \widehat{C}\| \le \left[\left(\frac{n}{n_0}\right)^{\log_2 18} (n_0^2 + 6n_0) - 6n \right] u \|A\| \, \|B\| + O(u^2). \tag{22.18}$$

Proof. The proof is analogous to that of Theorem 22.2, but more tedious. It suffices to analyse the computation of C_{12}, and the recurrence corresponding to (22.16) is

$$c_k = 18c_{k-1} + 89\cdot 2^{k-1}, \quad k > r, \quad c_r = 4^r. \qquad \square$$

Note that the bound for the Winograd–Strassen method has exponent $\log_2 18 \approx 4.170$ in place of $\log_2 12 \approx 3.585$ for Strassen's method, suggesting that the price to be paid for a reduction in the number of additions is an increased rate of error growth. All the comments above about Strassen's method apply to the Winograd variant.

Two further questions are suggested by the error analysis:

- How do the actual errors compare with the bounds?
- Which formulae are the more accurate in practice, Strassen's or Winograd's variant?

To give some insight we quote results obtained with a single precision Fortran 90 implementation of the two methods (the code is easy to write if we exploit the language's dynamic arrays and recursive procedures). We take random $n \times n$ matrices A and B and look at $\|AB - fl(AB)\|/(\|A\|\,\|B\|)$ for $n_0 = 1, 2, \dots, 2^k = n$ (note that this is not the relative error, since the denominator is $\|A\|\,\|B\|$ instead of $\|AB\|$, and note that no = n corresponds to conventional multiplication). Figure 22.2 plots the results for one random matrix of order 1024 from the uniform [0, 1] distribution and another matrix of the same size from the uniform [– 1, 1] distribution. The error bound (22.14) for Strassen's method is also plotted. Two observations can be made.

- Winograd's variant can be more accurate than Strassen's formulae, for all no, despite its larger error bound.
- The error bound overestimates the actual error by a factor up to 1.8×10^6 for n = 1024, but the variation of the errors with no is roughly as predicted by the bound.

22.2.3. Bilinear Noncommutative Algorithms

Bini and Lotti [97, 1980] have analysed the stability of bilinear noncommutative algorithms in general. They prove the following result.

Theorem 22.4 (Bini and Lotti). *Let $A, B \in \mathbb{R}^{n\times n}$ $(n = h^k)$ and let the product $C = AB$ be formed by a recursive application of the bilinear noncommutative algorithm (22.7), which multiplies $h \times h$ matrices using t nonscalar multiplications. The computed product $\widehat{C}$ satisfies*

$$\|C - \widehat{C}\| \le \alpha u n^{\log_h \beta} \log_h n \|A\|\,\|B\| + O(u^2), \tag{22.19}$$

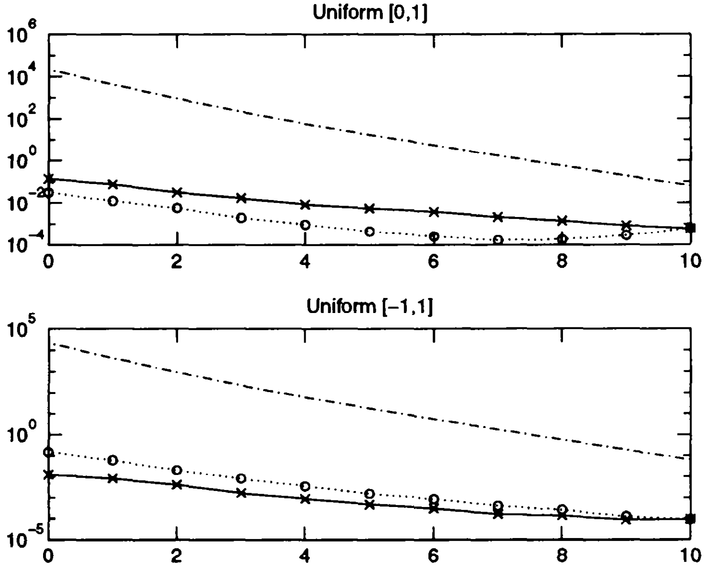


Figure 22.2. *Errors for Strassen's method with two random matrices of dimension* $n = 1024$. *Strassen's formulae:* "x", *Winograd's variant:* "o". *X-axis contains* $\log_2$ *of recursion threshold* n_0, $1 \le n_0 \le n$. *Dot-dash line is error bound for Strassen's formulae.*

where a and b are constants that depend on the number of nonzero terms in the matrices U, V and W that define the algorithm. □

The precise definition of α and b is given in [97, 1980]. If we take $k = 1$, so that $h = n$, and if the basic algorithm (22.7) is chosen to be conventional multiplication, then it turns out that $\alpha = n - 1$ and $b = n$, so the bound of the theorem becomes $(n - 1)nu\|A\|\,\|B\| + O(u^2)$, which is essentially the same as (22.17). For Strassen's method, $h = 2$ and $t = 7$, and $\alpha = 5$, $b = 12$, so the theorem produces the bound $5un^{\log_2 12}\log_2 n\|A\|\,\|B\| + O(u^2)$, which is a factor $\log_2 n$ larger than (22.14) (with $n_0 = 1$). This extra weakness of the bound is not surprising given the generality of Theorem 22.4.

Bini and Lotti consider the set of all bilinear noncommutative algorithms that form 2 x 2 products in 7 multiplications and that employ integer constants of the form $\pm 2^i$, where i is an integer (this set breaks into 26 equivalence classes). They show that Strassen's method has the minimum exponent b in its error bound in this class (namely, $b = 12$). In particular, Winograd's variant of Strassen's method has $b = 18$, so Bini and Lotti's bound has the same exponent $\log_2 18$ as in Theorem 22.3.

22.2.4. The 3M Method

A simple example reveals a fundamental weakness of the 3M method. Consider the computation of the scalar

$$z = x + iy = (\theta + i/\theta)^2 = \theta^2 - 1/\theta^2 + 2i.$$

In floating point arithmetic, if y is computed in the usual way, as $y = q(1/q) + (1/q)q$, then no cancellation occurs and the computed $\widehat{y}$ has high relative accuracy: $|\widehat{y} - y|/|y| = O(u)$. The 3M method computes

$$y = \Big(\theta + \frac{1}{\theta}\Big)\Big(\theta + \frac{1}{\theta}\Big) - \theta^2 - \frac{1}{\theta^2}.$$

If $|\theta|$ is large this formula expresses a number of order 1 as the difference of large numbers. The computed $\widehat{y}$ will almost certainly be contaminated by rounding errors of order uq^2, in which case the relative error is large: $|\widehat{y} - y|/|y| = O(u\theta^2)$. However, if We measure the error in $\widehat{y}$ relative to z, then it is acceptably small: $|\widehat{y} - y|/|z| = O(u)$. This example suggests that the 3M method may be stable, but in a weaker sense than for conventional multiplication.

To analyse the general case, consider the product $C_1 + iC_2 = (A_1 + iA_2)(B_1 + iB_2)$, where $A_k, B_k, C_k \in \mathbb{R}^{n\times n}$, $k = 1{:}2$. Using (22.10) we find that the computed product from conventional multiplication,

$$\widehat{C} = fl\big(A_1B_1 - A_2B_2 + i(A_1B_2 + A_2B_1)\big),$$

satisfies

$$|C_1 - \widehat{C}_1| \le (n+1)u\big(|A_1||B_1| + |A_2||B_2|\big) + O(u^2), \tag{22.20}$$

$$|C_2 - \widehat{C}_2| \le (n+1)u\big(|A_1||B_2| + |A_2||B_1|\big) + O(u^2). \tag{22.21}$$

For the 3M method C_1 is computed in the conventional way, and so (22.20) holds. It is straightforward to show that $\widehat{C}_2$ satisfies

$$|C_2 - \widehat{C}_2| \le (n+4)u\big[(|A_1| + |A_2|)(|B_1| + |B_2|) + |A_1||B_1| + |A_2||B_2|\big] + O(u^2). \tag{22.22}$$

Two notable features of the bound (22.22) are as follows. First, it is of a different and weaker form than (22.21); in fact, it exceeds the sum of the bounds (22.20) and (22.21). Second and more pleasing, it retains the property of (22.20) and (22.21) of being invariant under diagonal scalings

$$C = AB \rightarrow D_1AD_2 \cdot D_2^{-1}BD_3 = D_1CD_3, \qquad D_j \text{ diagonal,}$$

in the sense that the upper bound ΔC_2 in (22.22) scales also according to $D_1\Delta C_2D_3$. (The "hidden" second-order terms in (22.20)–(22.22) are invariant under these diagonal scalings.)

The disparity between (22.21) and (22.22) is, in part, a consequence of the differing numerical cancellation properties of the two methods. It is easy to show that there are always subtractions of like-signed numbers in the 3M method, whereas if A_1, A_2, B_l, and B_2 have nonnegative elements (for example) then no numerical cancellation takes place in conventional multiplication.

We can define a measure of stability with respect to which the 3M method matches conventional multiplication by taking norms in (22.21) and (22.22). We obtain the weaker bounds

$$\|C_2 - \widehat{C}_2\|_\infty \le 2(n+1)u\|A\|_\infty\|B\|_\infty + O(u^2), \tag{22.23}$$

$$\|C_2 - \widehat{C}_2\|_\infty \le 4(n+4)u\|A\|_\infty\|B\|_\infty + O(u^2) \tag{22.24}$$

(having used $\| \, |A_1| + |A_2| \, \| \le \sqrt{2}\, \|A_l + iA_2\|$). Combining these with an analogous weakening of (22.20), we find that for both conventional multiplication and the 3M method the computed complex matrix $\widehat{C}$ satisfies

$$\|C - \widehat{C}\|_\infty \le c_n u\|A\|_\infty\|B\|_\infty + O(u^2),$$

where $c_n = O(n)$.

The conclusion is that the 3M method produces a computed product $\widehat{C}$ whose imaginary part may be contaminated by relative errors much larger than those for conventional multiplication (or, equivalently, much larger than can be accounted for by small componentwise perturbations in the data A and B). However, if the errors are measured relative to $\|A\|\,\|B\|$, which is a natural quantity to use for comparison when employing matrix norms, then they are just as small as for conventional multiplication.

It is straightforward to show that if the 3M method is implemented using Strassen's method to form the real matrix products, then the computed complex product $\widehat{C}$ satisfies the same bound (22.14) as for Strassen's method itself, but with an extra constant multiplier of 6 and with 4 added to the expression inside the square brackets.

22.3. Notes and References

A good introduction to the construction of fast matrix multiplication methods is provided by the papers of Pan [816, 1984] and Laderman, Pan, and Sha [684, 1992].

Harter [504, 1972] shows that Winograd's formula (22.2) is the best of its kind, in a sense made precise in [504, 1972].

How does one derive formulae such as those of Winograd and Strassen, or that in the 3M method? Inspiration and ingenuity seem to be the key. A fairly straightforward, but still not obvious, derivation of Strassen's method is given by Yuval [1124, 1978]. Gustafson and Aluru [491, 1996] develop algorithms that systematically search for fast algorithms, taking advantage of a parallel computer. In an exhaustive search taking 21 hours of computation time on a 256 processor nCUBE 2, they were able to find 12 methods for multiplying 2 complex numbers in 3 multiplications and 5 additions; they could not find a method with fewer additions, thus proving that such a method does not exist. However, they estimate that a search for Strassen's method on a 1024 processor nCUBE 2 would take many centuries, even using aggressive pruning rules, so human ingenuity is not yet redundant!

To obtain a useful implementation of Strassen's method a number of issues have to be addressed, including how to program the recursion, how best to handle rectangular matrices of arbitrary dimension (since the basic method is defined only for square matrices of dimension a power of 2), and how to cater for the extra storage required by the method. These issues are discussed by Bailey [43, 1988], Bailey, Lee, and Simon [47, 1991], Fischer [374, 1974], Higham [544, 1990], Kreczmar [673, 1976], and Spieß [934, 1976], among others. Douglas, Heroux, Slishman, and Smith [317, 1994] present a portable Fortran implementation of Winograd's variant of Strassen's method for real and complex matrices, with a level-3 BLAS interface; they take care to use a minimal amount of extra storage (about $2n^3/3$ elements of extra storage is required when multiplying $n \times n$ matrices).

Higham [544, 1990] shows how Strassen's method can be used to produce algorithms for all the level-3 BLAS operations that are asymptotically faster than the conventional algorithms. Most of the standard algorithms in numerical linear algebra remain stable (in an appropriately weakened sense) when fast level-3 BLAS are used. See, for example, Chapter 12, $18.4, and Problems 11.4 and 13.2.

Knight [664, 1995] shows how to choose the recursion threshold to minimize the operation count of Strassen's method for rectangular matrices. He also shows how to use Strassen's method to compute the QR factorization of an $m \times n$ matrix in $O(mn^{1.838})$ operations instead of the usual $O(mn^2)$.

Bailey, Lee, and Simon [47, 1991] substituted their Strassen's method code for a conventionally coded BLAS3 subroutine `SGEMM` and tested LAPACK'S LU factorization subroutine `SGETRF` on a Cray Y-MP. They obtained speed improvements for matrix dimensions 1024 and larger.

The Fortran 90 standard includes an intrinsic function `MATMUL` that returns the product of its two matrix arguments. The standard does not specify which method is to be used for the multiplication. An IBM compiler supports the use of Winograd's variant of Strassen's method, via an optional third argument to `MATMUL` (an extension to Fortran 90) [318, 1994],

Brent was the first to point out the possible instability of Winograd's method [143, 1970]. He presented a full error analysis (including Theorem 22. 1) and showed that scaling ensures stability.

An error analysis of Strassen's method was given by Brent in 1970 in an unpublished technical report that has not been widely cited [142, 1970]. Section 22.2.2 is based on Higham [544, 1990].

According to Knuth, the 3M formula was suggested by P. Ungar in 1963 [668, 1981, p. 647]. It is analogous to a formula of Karatsuba and Ofman [643, 1963] for squaring a $2n$-digit number using three squarings of n-digit numbers. That three multiplications (or divisions) are *necessary* for evaluating the product of two complex numbers was proved by Winograd [1106, 1971].

Section 22.2.4 is based on Higham [552, 1992].

The answer to the question "What method should we use to multiply complex matrices?" depends on the desired accuracy and speed. In a Fortran environment an important factor affecting the speed is the relative efficiency of real and complex arithmetic, which depends on the compiler and the computer (complex arithmetic is automatically converted by the compiler into real arithmetic). For a discussion and some statistics see [552, 1992].

The efficiency of Winograd's method is very machine dependent. Bjørstad, Marine, Sørevik, and Vajteršic [122, 1992] found the method useful on the MasPar MP-1 parallel machine, on which floating point multiplication takes about three times as long as floating point addition at 64-bit precision. They also implemented Strassen's method on the MP-1 (using Winograd's method at the bottom level of recursion) and obtained significant speedups over conventional multiplication for dimensions as small as 256.

As noted in §22.1, Strassen [962, 1969] gave not only a method for multiplying n x n matrices in $O(n^{\log_2 7})$ operations, but also a method for inverting an n x n matrix with the same asymptotic cost. The method is described in Problem 22.8. For more on Strassen's inversion method see $24.3.2, Bailey and Ferguson [41, 1988], and Bane, Hansen, and Higham [51, 1993].

Problems

22.1. (Knight [664, 1995]) Suppose we have a method for multiplying n x n matrices in $O(n^\alpha)$operations, where $2 < \alpha < 3$. Show that if A is m x n and B is n x p then the product AB can be formed in $O(n_1^{\alpha-2}n_2n_3)$ operations, where n_l = min(m, n, p) and n_2 and n_3 are the other two dimensions.

22.2. Work out the operation count for Winograd's method applied to n x n matrices.

22.3. Let $S_n(n_0)$ denote the operation count (additions plus multiplications) for Strassen's method applied to n x n matrices, with recursion down to the level of no x no matrices. Assume that n and no are powers of 2. For large n, estimate $S_n(8)/S_n(n)$ and $S_n(1)/S_n(8)$ and explain the significance of these ratios (use (22.5)).

22.4. (Knight [664, 1995]) Suppose that Strassen's method is used to multiply an $m \times n$ matrix by an $n \times p$ matrix, where $m = a2^j$, $n = b2^j$, $p = c2^j$, and that conventional multiplication is used once any dimension is 2^r or less. Show that the operation count is $\alpha 7^j + b4^j$, where

$$\alpha = 2abc\left(\frac{8}{7}\right)^r + \frac{5}{3}(ab+bc+ac)\left(\frac{4}{7}\right)^r,$$
$$\beta = -\frac{1}{3}(5ab+5bc+8ac).$$

Show that by setting $r = 0$ and $a = 1$ a special case of the result of Problem 22.1 is obtained.

22.5. Compare and contrast Winograd's inner product formula for n = 2 with the imaginary part of the 3M formula (22.8).

22.6. Prove the error bound described at the end of §22.2.4 for the combination of the 3M method and Strassen's method.

22.7. Two fast ways to multiply complex matrices are (a) to apply the 3M method to the original matrices and to use Strassen's method to form the three real matrix products, and (b) to use Strassen's method with the 3M method applied at the bottom level of recursion. Investigate the merits of the two approaches with respect to speed and storage.

22.8. Strassen [962, 1969] gives a method for inverting an n x n matrix in $O(n^{\log_2 7})$ operations. Assume that n is even and write

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad A_{ij} \in \mathbb{R}^{m\times m}, \quad n = 2m.$$

The inversion method is based on the following formulae:

$$\begin{aligned} P_1 &= A_{11}^{-1}, & P_2 &= A_{21}P_1, \\ P_3 &= P_1A_{12}, & P_4 &= A_{21}P_3, \\ P_5 &= P_4 - A_{22}, & P_6 &= P_5^{-1}, \end{aligned}$$
$$A^{-1} = \begin{bmatrix} P_1 - P_3P_6P_2 & P_3P_6 \\ P_6P_2 & -P_6 \end{bmatrix}.$$

The matrix multiplications are done by Strassen's method and the inversions determining P_1 and P_6 are done by recursive invocations of the method itself. (a) Verify these formulae, using a block LU factorization of A, and show that they permit the claimed $O(n^{\log_2 7})$ complexity. (b) Show that if A is upper triangular, Strassen's method is equivalent to (the unstable) Method 2B of §13.2.2.

(For a numerical investigation into the stability of Strassen's inversion method, see §24.3.2.)

22.9. Find the inverse of the block upper triangular matrix

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix}.$$

Deduce that matrix multiplication can be reduced to matrix inversion.

22.10. (RESEARCH PROBLEM) Carry out extensive numerical experiments to test the accuracy of Strassen's method and Winograd's variant (cf. the results at the end of §22.2.2).

Chapter 23

The Fast Fourier Transform and Applications

Once the [FFT] method was established
it became clear that it had a long and interesting prehistory
going back as far as Gauss.
But until the advent of computing machines
it was a solution looking for a problem.
— T. W. KÖRNER, *Fourier Analysis (1988)*

Life as we know it would be very different without the FFT.
— CHARLES F. VAN LOAN, *Computation/*
Frameworks for the Fast Fourier Transform (1992)

23.1. The Fast Fourier Transform

The matrix–vector product $y = F_n x$, where

$$F_n = \Big(\exp(-2\pi i(r-1)(s-1)/n)\Big)_{r,s=1}^{n},$$

is the key computation in the numerical evaluation of Fourier transforms. If the product is formed in the obvious way then $O(n^2)$ operations are required. The fast Fourier transform (FFT) is a way to compute y in just $O(n \log n)$ operations. This represents a dramatic reduction in complexity.

The FFT is best understood (at least by a numerical analyst!) by interpreting it as the application of a clever factorization of the discrete Fourier transform (DFT) matrix F_n.

Theorem 23.1 (Cooley–Tukey radix 2 factorization). *If $n = 2^t$ then the DFT matrix F_n may be factorized as*

$$F_n = A_t \dots A_1 P_n, \tag{23.1}$$

where P_n is a permutation matrix and

$$\begin{aligned} A_k &= I_{2^{t-k}} \otimes B_{2^k}, \\ B_{2^k} &= \begin{bmatrix} I_r & \Omega_r \\ I_r & -\Omega_r \end{bmatrix}, \qquad r = 2^{k-1}, \\ \Omega_r &= \operatorname{diag}(1, \omega_k, \dots, \omega_k^{r-1}), \qquad \omega_k = \exp(-2\pi i/2^k). \end{aligned}$$

Proof. See Van Loan [1044, 1992, Thm. 1.3.3]. □

The theorem shows that we can write $y = F_n x$ as

$$y = A_t \dots A_1 P_n x.$$

which is formed as a sequence of matrix-vector products. It is the sparsity of the A_k (two nonzeros per row) that yields the $O(n \log n)$ operation count.

We will not consider the implementation of the FFT, and therefore we do not need to define the "bit reversing: permutation matrix P_n in (23.1). However, the way in which the weights ω_k^j are computed does affect the accuracy. We will assume that computed weights $\widehat{\omega}_k^j$ are used that satisfy, for all j and k,

$$\widehat{\omega}_k^j = \omega_k^j + \epsilon_{kj}, \qquad |\epsilon_{kj}| \le \mu. \tag{23.2}$$

Among the many methods for computing the weights are ones for which we can take $m = cu$, $m = cu \log j$, and $m = cuj$, where c is a constant that depends on the method; see Van Loan [1044, 1992, §1.4].

We are now ready to prove an error bound.

Theorem 23.2. *Let* $\widehat{y} = fl(F_n x)$, *where* $n = 2^t$, *be computed using the Cooley-Tukey radix* 2 *FFT, and assume that* (23.2) *holds. Then*

$$\frac{\|y-\widehat{y}\|_2}{\|y\|_2} \le n^{1/2}\frac{t\eta}{1-t\eta}, \qquad \eta := \mu + \gamma_4(1+\mu).$$

Proof. Denote by $\widehat{A}_k$ the matrix defined in terms of the computed $\widehat{\omega}_k^j$. Then

$$\begin{aligned}\widehat{y} &= fl(\widehat{A}_t \dots \widehat{A}_1 P_n x),\\ &= (\widehat{A}_t + \Delta\widehat{A}_t)\dots(\widehat{A}_1 + \Delta\widehat{A}_1)P_n x, \qquad |\Delta\widehat{A}_k| \le \gamma_4 |\widehat{A}_k|,\end{aligned}$$

using the fact that each A_k has only two nonzeros per row, and recalling that we are using complex arithmetic. In view of (23.2),

$$\widehat{A}_k = A_k + \Delta A_k, \qquad |\Delta A_k| \le \mu |A_k|.$$

Hence, overall,

$$\begin{aligned}\widehat{y} &= (A_t + E_t)\dots(A_1 + E_1)P_n x,\\ |E_k| &\le \big(\mu + \gamma_4(1+\mu)\big)|A_k| = \eta |A_k|.\end{aligned}$$

Invoking Lemma 3.7 we find that

$$\begin{aligned}|y-\widehat{y}| &\le \big[(1+\eta)^t - 1\big]|A_t|\dots|A_1||P_n||x|\\ &\le \frac{t\eta}{1-t\eta}|A_t|\dots|A_1||P_n||x|,\end{aligned}$$

using Lemma 3.1 for the second inequality. Now $\|\,|A_k|\,\|_2 = \|\,|B_{2^k}|\,\|_2 = \|\big[\begin{smallmatrix} I & I\\ I & I\end{smallmatrix}\big]\|_2 = 2$. Hence

$$\|y-\widehat{y}\|_2 \le \frac{t\eta}{1-t\eta}2^t\|x\|_2. \tag{23.3}$$

Finally, because F_n is $\sqrt{n}$ times a unitary matrix ($F_n^* F_n = nI$), $\|x\|_2 = n^{-1/2}\|y\|_2$. ☐

Theorem 23.2 says that the Cooley–Tukey radix 2 FFT yields a tiny forward error, provided that the weights are computed stably. It follows immediately that the computation is backward stable, since $\widehat{y} = y + \Delta y = F_n x + \Delta y$ implies $\widehat{y} = F_n(x + \Delta x)$ with $\|\Delta x\|_2/\|x\|_2 = \|\Delta y\|_2/\|y\|_2$. If we form $y = F_n x$ by conventional multiplication using the exact F_n, then (Problem 3.7) $|y-\widehat{y}| \le \gamma_{n+2}|F_n||x|$, so $\|y-\widehat{y}\|_2 \le n^{1/2}\gamma_{n+2}\|y\|_2$. Hence when *m* is of order *u*, the FFT has an error bound smaller than that for conventional multiplication by the same factor as the reduction in complexity of the method. To sum up, the FFT is perfectly stable.

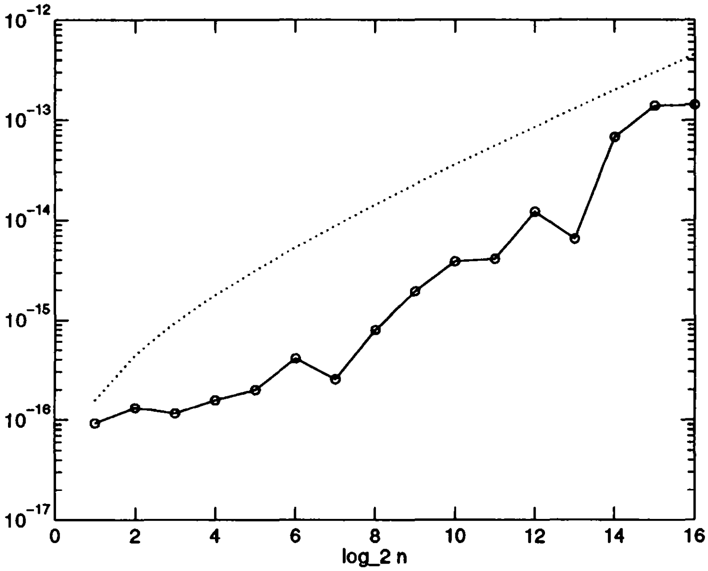


Figure 23.1. *Error in FFT followed by inverse FFT* ("o"). *Dotted line is error bound.*

The inverse transform $x = F_n^{-1}y = n^{-1}F_n^*y$ can again be computed in $O(n \log n)$ operations using the Cooley–Tukey radix 2 factorization, and $\|x - \widehat{x}\|_2/\|x\|_2$ satisfies the same bound as in Theorem 23.2. (Strictly, we should replace t by $t + 1$ in the bound to account for the rounding error in dividing by n)

For other variations of the FFT, based on different radices or different factorization of F_n, results analogous to Theorem 23.2 hold.

A simple way to test the error bounds is to compute the FFT followed by the inverse transform, $\widehat{x} = fl(F_n^{-1} fl(F_n x))$, and to evaluate $e_n = \|x - \widehat{x}\|_2/\|x\|_2$. Our analysis gives the bound $e_n \le 2n^{1/2} \log_2 nh + O(h^2)$. Figure 23.1 plots e_n and the approximate error bound $n^{1/2} \log_2 nu$ for $n = 2^k$, $k = 0{:}16$, with random x from the normal $N(0,1)$ distribution (the FFTs were computed using MATLAB's `fft` and `ifft` functions). The errors grow at roughly the same rate as the bound and are on average about a factor of 10 smaller than the bound.

23.2. Circulant Linear Systems

A circulant matrix (or circulant, for short) is a special Toeplitz matrix in which the diagonals wrap around:

$$C = \begin{bmatrix} c_1 & c_n & \dots & c_2 \\ c_2 & c_1 & \dots & \vdots \\ \vdots & \ddots & \ddots & c_n \\ c_n & \dots & c_2 & c_1 \end{bmatrix} \in \mathbb{C}^{n\times n}.$$

Circulant matrices have the important property that they are diagonalized by the DFT matrix F_n:

$$F_n C F_n^{-1} = D = \operatorname{diag}(d_i).$$

Moreover, the eigenvalues are given by $d = F_n c$. Hence a linear system $Cx = b$ can be solved in $O(n \log n)$ operations with the aid of the FFT as follows:

(1) $d = F_n c$,
(2) $g = F_n b$,
(3) $h = D^{-1} g$,
(4) $x = F_n^{-1} h$.

The computation involves two FFTs, a diagonal scaling, and an inverse FFT. We now analyse the effect of rounding errors. It is convenient to write the result of Theorem 23.2 in the equivalent form (from (23.3))

$$\widehat{y} = (F_n + \Delta F_n)x, \qquad \|\Delta F_n\|_2 \le n\frac{t\eta}{1-t\eta} =: f(n,u). \tag{23.4}$$

Steps (1) and (2) yield

$$\begin{aligned} \widehat{d} &= (F_n + \Delta_1)c, \qquad &\|\Delta_1\|_2 \le f(n,u), \\ \widehat{g} &= (F_n + \Delta_2)b, \qquad &\|\Delta_2\|_2 \le f(n,u). \end{aligned} \tag{23.5}$$

Equation (23.5) implies that $\widehat{D} = D + \Delta_0$, $\|\Delta_0\|_2 \le f(n,u)\|C\|_2$.

For steps (3) and (4) we have, using Lemma 3.5,

$$\begin{aligned} \widehat{h} &= (\widehat{D} + \Delta D)^{-1}\widehat{g}, \qquad &|\Delta D| \le \sqrt{2}\gamma_4|\widehat{D}|, \\ \widehat{x} &= (F_n^{-1} + \Delta_3)\widehat{h}, \qquad &\|\Delta_3\|_2 \le n^{-1}f(n,u). \end{aligned}$$

Putting these equations together we have

$$\widehat{x} = (F_n^{-1} + \Delta_3)(\widehat{D} + \Delta D)^{-1}(F_n + \Delta_2)b, \tag{23.6}$$

or, working to first order,

$$\begin{aligned} b &= (F_n + \Delta_2)^{-1}(\widehat{D} + \Delta D)(F_n^{-1} + \Delta_3)^{-1}\widehat{x} \\ &\approx (F_n^{-1} - F_n^{-1}\Delta_2 F_n^{-1})(\widehat{D} + \Delta D)(F_n - F_n\Delta_3 F_n)\widehat{x} \\ &\approx (F_n^{-1}\widehat{D}F_n + F_n^{-1}\Delta D F_n - F_n^{-1}\Delta_2 F_n^{-1}\widehat{D}F_n - F_n^{-1}\widehat{D}F_n\Delta_3 F_n)\widehat{x} \\ &=: (C + \Delta C)\widehat{x}. \end{aligned}$$

We obtain a backward stability result.

Theorem 23.3. *Let $C \in \mathbb{C}^{n\times n}$ be a circulant and suppose the system $Cx = b$ is solved by the FFT process described above, where the FFT satisfies* (23.4). *Then the computed solution $\widehat{x}$ satisfies $(C + \Delta C)\widehat{x} = b$, where*

$$\|\Delta C\|_2 \le \bigl((1+2n^{-1/2})f(n,u)+4\sqrt{2}u\bigr)\|C\|_2 + O(u^2). \qquad \square$$

The conclusion is that the FFT method for solving a circulant system is normwise backward stable provided that the FFT itself is computed stably.

Although we have shown that $\widehat{x}$ solves a slightly perturbed system, the perturbed matrix $C + \Delta C$ is not a circulant. In fact, $\widehat{x}$ does not, in general, solve a nearby circulant system, as can be verified experimentally by computing the "(circulant backward error” using techniques from [527, 1992]. The basic reason for the lack of this stronger form of stability is that there are too few parameters in the matrix onto which to throw the backward error.

A forward error bound can be obtained by applying standard perturbation theory to Theorem 23.3: the forward error is bounded by a multiple of $\kappa_2(C)u$. That the forward error can be as large as $\kappa_2(C)u$ is clear from the analysis above, because (23.5) shows that the computed eigenvalue $\min_i|\widehat{d}_i|$ is contaminated by an error of order $u\max_i|\widehat{d}_i|$.

23.3. Notes and References

For a unified treatment of the many different versions of the FFT, including implementation details, see Van Loan [1044, 1992].

For a comprehensive survey of the discrete Fourier transform see Briggs and Henson [148, 1995].

The Cooley–Tukey radix 2 FFT was presented in [240, 1965], which is one of the most cited mathematics papers of all time [554, 1993, p. 171].

The history of the FFT is discussed by Cooley [238, 1990], [239, 1994] and Cooley and Tukey [241, 1993]. Cooley [238, 1990] states that the earliest known reference to the FFT is an obscure 1866 paper of Gauss in neoclassic Latin, and he recommends that researchers not publish papers in neoclassic Latin!

Theorem 23.2 is not new, but the proof is more concise than most in the literature. Early references that derive error bounds using the matrix factorization formulation of the FFT are Gentleman and Sande [434, 1966] and Ramos [859, 1971]. A full list of references for error analysis of the FFT is given by Van Loan [1044, 1992, §1.4].

Linzer [708, 1992] shows that the FFT-based circulant solver is forward stable and poses the question of whether or not the solver is backward stable. Our backward error analysis answers this question positively and therefore also proves forward stability.

One application of circulant linear systems is in the preconditioned conjugate gradient method for solving Toeplitz systems. The idea of using a circulant preconditioned was suggested by Strang [960, 1986], and the theory and practice of this technique is now well developed. For more details see Chan, Nagy, and Plemmons [191, 1994] and the references therein. A good source of results about circulant matrices is the book by Davis [266, 1979].

Problems

23.1. (Bailey [44, 1993]) In high-precision multiplication we have two integer n-vectors x and y representing high-precision numbers and we wish to form the terms $z_k = \sum_{j=1}^{k} x_j y_{k+1-j}$, $k = 1\colon n$. By padding the vectors with n zeros, these products can be expressed in the form $z_k = \sum_{j=1}^{2n} x_j y_{k+1-j}$, where $k + 1 - j$ is interpreted as $k + 1 - j + n$ if $k + 1 - j$ is negative. These products represent a convolution: a matrix–vector product involving a circulant matrix. Analogously to the linear system solution in §23.2, this product can be evaluated in terms of discrete Fourier transforms as $z = F_n^{-1}(F_n x \cdot F_n y)$, where the dot denotes componentwise multiplication of two vectors. Since x and y are integer vectors, the z_i should also be integers, but in practice they will be contaminated by rounding errors. Obtain a bound on $z - \widehat{z}$ and deduce a sufficient condition for the “nearest integer vector to $\widehat{z}$ to be the exact z.

Chapter 24

Automatic Error Analysis

Given the pace of technology,
I propose we leave math to the machines and go play outside.
— CALVIN, *Calvin and Hobbes by* Bill Watterson (1992)

To analyse a given numerical algorithm we proceed as follows.
A number which measures the effect of roundoff error
is assigned to each set of data.
"Hill-climbing" procedures are then applied to search for
values large enough to signal instability.
— WEBB MILLER, *Software for Roundoff Analysis (1975)*

Automatic error analysis is any process in which we use the computer to help us analyse the accuracy or stability of a numerical computation. The idea of automatic error analysis goes back to the dawn of scientific computing. For example, running error analysis, described in §3.3, is a form of automatic error analysis; it was used extensively by Wilkinson on the ACE machine. Various forms of automatic error analysis have been developed. In this chapter we describe in detail the use of direct search optimization for investigating questions about the stability and accuracy of algorithms. We also describe interval analysis and survey other forms of automatic error analysis.

24.1. Exploiting Direct Search Optimization

Is Algorithm X numerically stable? How large can the growth factor be for Gaussian elimination (GE) with pivoting strategy P? By how much can condition estimator C underestimate the condition number of a matrix? These types of questions are common, as we have seen in this book. Usually, we attempt to answer such questions by a combination of theoretical analysis and numerical experiments with random and nonrandom data. But a third approach can be a valuable supplement to the first two: phrase the question as an optimization problem and apply a direct search method.

A direct search method for the problem

$$\max_{x\in\mathbb{R}^n} f(x), \qquad f:\mathbb{R}^n \to \mathbb{R} \tag{24.1}$$

is a numerical method that attempts to locate a maximizing point using function values only and does not use or approximate derivatives of *f*. Such methods are usually based on heuristics that do not involve assumptions about the function *f*. Various direct search methods have been developed; for surveys see Powell [838, 1970] and Swarm [978, 1972], [979, 1974]. Most of these methods were developed in the 1960s, in the early years of numerical optimization. For problems in which *f* is smooth, direct search methods have largely been supplanted by more sophisticated optimization methods that use derivatives (such as quasi-Newton methods and conjugate gradient methods), but they continue to find use in applications where *f* is not differentiable, or even not continuous. These applications range from chemical analysis [881, 1977], where direct search methods have found considerable use, to the determination of drug doses in the treatment of cancer [93, 1991]; in both applications the evaluation of *f* is affected by experimental errors. Lack of smoothness of *f*, and the difficulty of obtaining derivatives when they exist, are characteristic of the optimization problems we consider here.

The use of direct search can be illustrated with the example of the growth factor for GE on $A \in \mathbb{R}^{n\times n}$,

$$\rho_n(A) = \frac{\max_{i,j,k} |a_{ij}^{(k)}|}{\max_{i,j} |a_{ij}|},$$

where the $a_{ij}^{(k)}$ are the intermediate elements generated during the elimination. We know from §9.2 that the growth factor governs the stability of GE, so for a given pivoting strategy we would like to know how big r_n *(A)* can be.

To obtain an optimization problem of the form (24.1) we let x = *vet(A)* $\in \mathbb{R}^{n^2}$, and we define *f(x)* := $r_n(A)$. Then we wish to determine

$$\max_{x\in\mathbb{R}^{n^2}} f(x) \equiv \max_{A\in\mathbb{R}^{n\times n}} \rho_n(A).$$

Suppose, first, that no pivoting is done. Then *f* is defined and continuous at all points where the elimination does not break down, and it is differentiable except at points where there is a tie for the maximum in the numerator or denominator of the expression defining $r_n(A)$. We took n = 4 and applied the direct search maximizer MDS (described in §24.2) to *f(x)*, starting with the identity matrix $A = I_4$. After 11 iterations and 433 function evaluations, the maximizer converged[18], having located the matrix[19]

$$B = \begin{bmatrix} 1.7846 & -0.2760 & -0.2760 & -0.2760 \\ -3.3848 & 0.7240 & -0.3492 & -0.2760 \\ -0.2760 & -0.2760 & 1.4311 & -0.2760 \\ -0.2760 & -0.2760 & -0.2760 & 0.7240 \end{bmatrix},$$

for which $r_4(B)$ = 1.23 x 10^5. (The large growth is a consequence of the submatrix B(1:3, 1:3) being ill conditioned; B itself is well conditioned.) Thus the optimizer readily shows that $r_n(A)$ can be very large for GE without pivoting.

Next, consider GE with partial pivoting (GEPP). Because the elimination cannot break down, *f* is now defined everywhere, but it is usually discontinuous when there is a tie in the choice of pivot element, because then an arbitrarily small change in *A can* alter the pivot sequence. We applied the maximizer MDS to *f*, this time starting with the orthogonal matrix[20] $A \in \mathbb{R}^{4\times 4}$ with $a_{ij} = (2/\sqrt{2n+1})\sin(2ij\pi/(2n+1))$ (cf. (9.11)), for which $r_4(A)$ = 2.32. After 29 iterations and 1169 function evaluations the maximizer converged to a matrix B with $r_4(B)$ = 5.86. We used this matrix to start the maximizer

---

[18] In the optimizations of this section we used the convergence tests described in §24.2 with tol = 10^{-3}. There is no guarantee that when convergence is achieved it is to a local maximum; see §24.2.

[19] All numbers quoted are rounded to the number of significant figures shown.

[20] This matrix is `orthog (n, 2)` from the Test Matrix Toolbox; see Appendix E.

AD (described in §24.2); it took 5 iterations and 403 function evaluations to converge to the matrix

$$C = \begin{bmatrix} 0.7248 & 0.7510 & 0.5241 & 0.7510 \\ 0.7317 & 0.1889 & 0.0227 & -0.7510 \\ 0.7298 & -0.3756 & 0.1150 & 0.7511 \\ -0.6993 & -0.7444 & 0.6647 & -0.7500 \end{bmatrix},$$

for which $r_4(C) = 7.939$. This is one of the matrices described in Theorem 9.6, modulo the convergence tolerance.

These examples, and others presented below, illustrate the following attractions of using direct search methods to investigate the stability of a numerical computation.

(1) The simplest possible formulation of optimization problem is often sufficient to yield useful results. Derivatives are not needed, and direct search methods tend to be insensitive to lack of smoothness in the objective function f. Unboundedness of f is a favorable property-direct search methods usually quickly locate large values of f.

(2) Good progress can often be made from simple starting values, such as an identity matrix. However, prior knowledge of the problem may provide a good starting value that can be substantially improved (as in the partial pivoting example).

(3) Usually it is the global maximum of f in (24.1) that is desired (although it is often sufficient to know that f can exceed a specified value). When a direct search method converges it will, in general, at best have located a *local* maximum-and in practice the maximizer may simply have stagnated, particularly if a slack convergence tolerance is used. However, further progress can often be made by *restarting* the same (or a different) maximizer, as in the partial pivoting example. This is because for methods that employ a simplex (such as the MDS method), the behaviour of the method starting at x_0 is determined not just by x_0. but also by the $n + 1$ vectors in the initial simplex constructed at x_0.

(4) The numerical information revealed by direct search provides a starting point for further theoretical analysis. For example, the GE experiments above strongly suggest the (well known) results that $r_n(A)$ is unbounded without pivoting and bounded by 2^{n-1} for partial pivoting, and inspection of the numerical data suggests the methods of proof.

When applied to smooth problems the main disadvantages of direct search methods are that they have at best a linear rate of convergence and they are unable to determine the nature of the point at which they terminate (since derivatives are not calculated). These disadvantages are less significant for the problems we consider, where it is not necessary to locate a maximum to high accuracy and objective functions are usually nonsmooth. (Note that these disadvantages are not necessarily shared by methods that implicitly or explicitly *estimate* derivatives using function values, such as methods based on conjugate directions, for which see Powell [838, 1970], [839, 1975]; however, these are not normally regarded as direct search methods.)

A final attraction of direct search is that it can be used to test the correctness of an *implementation* of a stable algorithm. The software in question can be used in its original form and does not have to be translated into some other representation.

24.2. Direct Search Methods

For several years I have been using MATLAB implementations of three direct search methods. The first is the alternating directions (AD) method (also known as the coordinate search method). Given a starting value x it attempts to solve the problem (24.1) by repeatedly maximizing over each coordinate direction in turn:

```
repeat
    % One iteration comprises a loop over all components of $x$.
    for $i = 1:n$
        find $\alpha$ such that $f(x + \alpha e_i)$ is maximized (line search)
        Set $x \leftarrow x + \alpha e_i$
    end
until converged
```

AD is one of the simplest of all optimization methods and the fundamental weakness that it ignores any interactions between the variables is well known. Despite the poor reputation of AD we have found that it can perform well on the types of problems considered here. In our MATLAB implementation of AD the line search is done using a crude scheme that begins by evaluating $f(x + he_i)$ with $h = 10^{-4}x_i$ (or $h = 10^{-4}\max(\|x\|_\infty, 1)$ if $x_i = 0$); if $f(x + he_i) \le f(x)$ then the sign of h is reversed. Then if $f(x + he_i) > f(x)$, h is doubled at most 25 times until no further increase in f is obtained. Our convergence test checks for a sufficient relative increase in f between one iteration and the next: convergence is declared when

$$f_k - f_{k-1} \le \mathrm{tol}|f_{k-1}|, \tag{24.2}$$

where f_k is the highest function value at the end of the kth iteration. The AD method has the very modest storage requirement of just a single n-vector.

The second method is the multidirectional search method (MDS) of Dennis and Torczon [1008, 1989], [1009, 1991], [301, 1991]. This method employs a simplex, which is defined by $n + 1$ vectors $\{v_i\}_0^n$ in $\mathbb{R}^n$. One iteration in the case $n = 2$ is represented pictorially in Figure 24.1, and may be explained as follows.

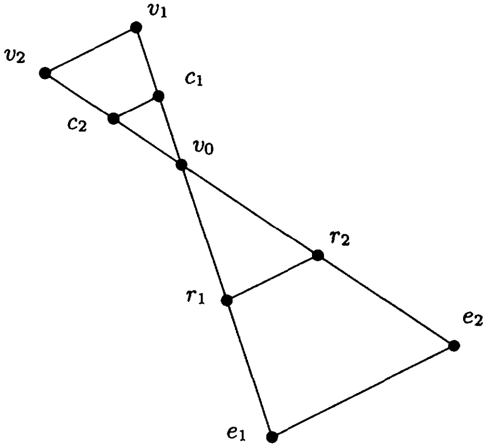


Figure 24.1. *The possible steps in one iteration of the MDS method when $n = 2$.*

The initial simplex is $\{v_0, v_1, v_2\}$ and it is assumed that $f(v_0) = \max_i f(v_i)$. The purpose of an iteration is to produce a new simplex at one of whose vertices f exceeds $f(v_0)$. In the first step the vertices v_1 and v_2 are reflected about v_0. along the lines joining them to v_0, yielding r_l and r_2 and the reflected simplex $\{v_0,r_1,r_2\}$. If this reflection step is successful, that is, if $\max_i f(r_i) > f(v_0)$, then the edges from v_0 to r_i are doubled in length to give an expanded simplex $\{v_0, e_l, e_2\}$. The original simplex is then replaced by $\{v_0, e_l, e_2\}$ if $\max_i f(e_i) > \max_i f(r_i)$, and otherwise by $\{v_0, r_1, r_2\}$. If the reflection step is unsuccessful then the edges $v_0 - v_i$ of the original simplex are shrunk to half their length to give the contracted simplex $\{v_0, c_1, c_2\}$. This becomes the new simplex if $\max_i f(c_i) > \max_i f(v_i)$, in which case the current iteration is complete; otherwise the algorithm jumps back to the reflection step, now working with the contracted simplex. For further details of the MDS method see Dennis and Torczon [301, 1991] and Torczon [1008, 1989], [1009, 1991].

The MDS method requires at least $2n$ independent function evaluations per iteration, which makes it very suitable for parallel implementation. Generalizations of the MDS method that are even more suitable for parallel computation are described in [301, 1991] and [1010, 1992]. The MDS method requires $O(n^2)$ elements of storage for the simplices, but this can be reduced to $O(n)$ (at the cost of extra bookkeeping) if an appropriate choice of initial simplex is made [301, 1991].

Unlike most direct search methods, the MDS method possesses some convergence theory. Torczon [1009, 1991] shows that if the level set of j at v_0^0

is compact and f is continuously differentiable on this level set then a subsequence of the points v_0^k (where k denotes the iteration index) converges to a stationary point of f. Moreover, she gives an extension of this result that requires only continuity of f and guarantees convergence to either a stationary point off or a point where f is not continuously differentiable.

Our implementation of the MDS method provides two possible starting simplices, both of which include the starting point x_0: a regular one (all sides of equal length) and a right-angled one based on the coordinate axes, both as described by Torczon in [1008, 1989]. The scaling is such that each edge of the regular simplex, or each edge of the right-angled simplex that is joined to x_0, has length $\max(\|x_0\|_\infty, 1)$. Also as in [1008, 1989], the main termination test halts the computation when the relative size of the simplex is no larger than a tolerance tol_1 that is, when

$$\frac{1}{\max(1, \|v_0\|_1)} \max_{1\le i\le n} \|v_i - v_0\|_1 \le \mathrm{tol}. \qquad (24.3)$$

Unless otherwise stated, we used tol = 10^{-3} in (24.2) and (24.3) in all our experiments.

The third method that we have used is the Nelder–Mead direct search method [787, 1965], [303, 1987], which also employs a simplex but which is fundamentally different from the MDS method. We omit a description since the method is described in textbooks (see, for example, Gill, Murray, and Wright [447, 1981, §4.2.2], or Press, Teukolsky, Vetterling, and Flannery [842, 1992, §10.4]). Our limited experiments with the Nelder-Mead method indicate that while it can sometimes out-perform the MDS method, the MDS method is generally superior for our purposes. No convergence results of the form described above for the MDS method are known for the Nelder-Mead method.

For general results on the convergence of "pattern search" methods, see Torczon [1011, 1993]. The AD and MDS methods are pattern search methods, but the Nelder–Mead method is not.

It is interesting to note that the MDS method, the Nelder-Mead method, and our particular implementation of the AD method do not exploit the numerical values of f: their only use of f is to compare two function values to see which is the larger!

Our MATLAB implementations of the AD, MDS, and Nelder-Mead direct search methods are in the Test Matrix Toolbox, described in Appendix E.

24.3. Examples of Direct Search

In this section we give examples of the use of direct search to investigate the behaviour of numerical algorithms.

24.3.1. Condition Estimation

We have experimented with MATLAB implementations of two matrix condition number estimators. RCOND is the LINPACK estimator, described in §14.4, as implemented in the built-in MATLAB function `rcond`. LACON is the LAPACK condition estimator, as given in Algorithm 14.4 and implemented in MATLAB'S `condest`. Both estimators compute a lower bound for $\kappa_1(A)$ by estimating $\|A^{-1}\|_1$ ($\|A\|_1$ is computed explicitly as the maximum column sum).

To put the problem in the form of (24.1), we define $x = \mathrm{vec}(A)$, $A \in \mathbb{R}^{n\times n}$, and

$$f(x) = \frac{\kappa_1(A)}{\mathrm{est}(A)},$$

where $\mathrm{est}(A) \le \kappa_1(A)$ is the condition estimate. We note that, since the algorithms underlying RCOND and LACON contain tests and branches, there are matrices A for which an arbitrarily small change in A can completely change the condition estimate; hence for both algorithms f has points of discontinuity.

We applied the MDS maximizer to RCOND starting at the 5 x 5 Hilbert matrix. After 67 iterations and 4026 function evaluations the maximizer had located a matrix for which $f(x) = 226.9$. We then started the Nelder-Mead method from this matrix; after another 4947 function evaluations it had reached the matrix (shown to 5 significant figures)

$$A = \begin{bmatrix} 3.6480 & 2.0412 & 7.6441 & 1.7729 & -2.8260 \\ -1.0542\times 10^{1} & 1.9044 & 1.7709 & 1.7755 & 1.7444 \\ -1.2110 & 1.8110 & 1.7027 & 1.7101 & -1.1567\times 10^{1} \\ 7.7564 & 1.7047 & 1.6479 & 1.6306 & 1.6459 \\ 3.1566\times 10^{1} & 1.7337 & 2.2495 & 1.7003 & 1.3812 \end{bmatrix},$$

for which

$$\kappa_1(A) = 3.38\times 10^{5}, \quad \mathrm{est}(A) = 1.65\times 10^{1}, \quad \frac{\kappa_1(A)}{\mathrm{est}(A)} = 2.05\times 10^{4}.$$

This example is interesting because the matrix is well scaled, while the parametrized counterexamples of Cline and Rew [217, 1983] all become badly scaled when the parameter is chosen to make the condition estimate poor.

For LACON we took as starting matrix the 4 x 4 version of then x n matrix with $a_{ij} = \cos((i-1)(j-1)\pi/(n-1))$ (this is a Chebyshev–Vandermonde matrix, as used in §21.3.3, and is `orthog(n, - 1)` in the Test Matrix Toolbox). After 11 iterations and 1001 function evaluations the AD maximizer had determined a (well-scaled) matrix A for which

$$\kappa_1(A) = 2.94\times 10^{5}, \quad \mathrm{est}(A) = 4.81, \quad \frac{\kappa_1(A)}{\mathrm{est}(A)} = 6.11\times 10^{4}.$$

With relatively little effort on our part (most of the effort was spent experimenting with different starting matrices), the maximizers have discovered examples where both condition estimators fail to achieve their objective of producing an estimate correct to within an order of magnitude. The value of direct search maximization in this context is clear: it can readily demonstrate the fallibility of a condition estimator—a task that can be extremely difficult to accomplish using theoretical analysis or tests with random matrices. Moreover, the numerical examples obtained from direct search may provide a starting point for the construction of parametrized theoretical ones, or for the improvement of a condition estimation algorithm.

In addition to measuring the quality of a single algorithm, direct search can be used to compare two competing algorithms to investigate whether one algorithm performs uniformly better than the other. We applied the MDS maximizer to the function

$$f(x) = \frac{\mathrm{estL}(A)}{\mathrm{estR}(A)},$$

where $\mathrm{estL}(A)$ and $\mathrm{estR}(A)$ are the condition estimates from LACON and RCOND, respectively. If $f(x) > 1$ then LACON has produced a larger lower bound for $\kappa_1(A)$ than RCOND. Starting with a random 5 x 5 matrix the Nelder–Mead maximizer produced after 1788 function evaluations a matrix A for which $\mathrm{estL}(A) = \kappa_1(A)$ and $f(x) = 1675.4$. With f defined as $f(x) = \mathrm{estR}(A)/\mathrm{estL}(A)$, and starting with I_4, after 6065 function evaluations the MDS maximizer produced a matrix for which $f(x) = 120.8$. This experiment shows that neither estimator is uniformly superior to the other. This conclusion would be onerous to reach by theoretical analysis of the algorithms.

24.3.2. Fast Matrix Inversion

We recall Strassen's inversion method from Problem 22.8: for

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbb{R}^{n\times n}, \quad A_{ij} \in \mathbb{R}^{m\times m}, \quad n = 2m,$$

it uses the formulae

$$\begin{aligned} P_1 &= A_{11}^{-1}, & P_2 &= A_{21}P_1, \\ P_3 &= P_1A_{12}, & P_4 &= A_{21}P_3, \\ P_5 &= P_4 - A_{22}, & P_6 &= P_5^{-1}, \end{aligned}$$

$$A^{-1} = \begin{bmatrix} P_1 - P_3P_6P_2 & P_3P_6 \\ P_6P_2 & -P_6 \end{bmatrix},$$

where each of the matrix products is formed using Strassen's fast matrix multiplication method. Strassen's inversion method is clearly unstable for general A, because the method breaks down if A_{11} is singular. Indeed Strassen's inversion method has been implemented on a Cray-2 by Bailey and Ferguson [41, 1988] and tested for $n \leq 2048$, and these authors observe empirically that the method has poor numerical stability. Direct search can be used to gain insight into the numerical stability.

With x = vet$(A) \in \mathbb{R}^{n^2}$, define the stability measure

$$f(x) = \frac{\min\{\|A\widehat{X} - I\|_\infty, \|\widehat{X}A - I\|_\infty\}}{\|A\|_\infty \|\widehat{X}\|_\infty}, \tag{24.4}$$

where $\widehat{X}$ is the inverse of A computed using Strassen's inversion method. This definition of f is appropriate because, as shown in Chapter 13, for most conventional matrix inversion methods either the left residual $XA - I$ or the right residual $AX - I$ is guaranteed to have norm of order $u\|X\| \, \|A\|$. To treat Strassen's inversion method as favorably as possible we use just one level of recursion; thus P_1 and P_6 are computed using GEPP but the multiplications are done with Strassen's method. We applied the MDS maximizer, with tol = 10^{-9} in (24.3), starting with the 4 x 4 Vandermonde matrix whose (i,j) element is $((j-1)/3)^{i-1}$. After 34 iterations the maximizer had converged with $f = 0.838$, which represents complete instability. The corresponding matrix A is well conditioned, with $_{k2}(A) = 82.4$. For comparison, the value of f when A is inverted using Strassen's method with *conventional* multiplication is $f = 6.90 \times 10^{-2}$; this confirms that the instability is not due to the use of fast multiplication techniques—it is inherent in the inversion formulae.

If A is a symmetric positive definite matrix then its leading principal submatrices are no more ill conditioned than the matrix itself, so we might expect Strassen's inversion method to be stable for such matrices. To investigate this possibility we carried out the same maximization as before, except we enforced positive definiteness as follows: when the maximizer generates a vector $x \equiv$ vec(B), A in (24.4) is defined as $A = B^TB$. Starting with a 4 x 4 random matrix A with $_{k2}(A) = 6.71 \times 10^7$, the maximization yielded the value $f = 3.32 \times 10^{-8}$ after 15 iterations, and the corresponding value of f when conventional multiplication is used is $f = 6.61 \times 10^{-11}$ (the "maximizing" matrix A has condition number $_{k2}(A) = 3.58 \times 10^9$).

The conclusion from these experiments is that Strassen's inversion method cannot be guaranteed to produce a small left or right residual even when A is symmetric positive definite and conventional multiplication is used. Hence the method must be regarded as being fundamentally unstable.

24.3.3. Solving a Cubic

Explicit formulae can be obtained for the roots of a cubic equation using techniques associated with the 16th century mathematicians del Ferro, Cardano, Tartaglia, and Vieta [140, 1968], [332, 1990]. The following development is based on Birkhoff and Mac Lane [102, 1977, §5.5].

Any nondegenerate cubic equation can be put in the form $x^3 + ax^2 + bx + c = O$ by dividing through by the leading coefficient. We will assume that the coefficients are real. The change of variable $x = y - a/3$ eliminates the quadratic term:

$$y^3 + py + q = 0, \qquad p = -\frac{a^2}{3} + b, \quad q = \frac{2}{27}a^3 - \frac{ab}{3} + c.$$

Then Vieta's substitution $y = w - p/(3w)$ yields

$$w^3 - \frac{p^3}{27w^3} + q = 0$$

and hence a quadratic equation in w^3: $(w^3)^2 + qw^3 — p^3/27 = 0$. Hence

$$w^3 = -\frac{q}{2} \pm \sqrt{\frac{q^2}{4} + \frac{p^3}{27}}. \tag{24.5}$$

For either choice of sign, the three cube roots for w yield the roots of the original cubic, on transforming back from w to y to z.

Are these formulae for solving a cubic numerically stable? Direct search provides an easy way to investigate. The variables are the three coefficients a, b, c and for the objective function we take an approximation to the relative error of the computed roots $\widehat{z} \in \mathbb{R}^3$. We compute the "exact" roots z using MATLAB's roots function (which uses the QR algorithm to compute the eigenvalues of the companion matrix[21]) and then our relative error measure is $\min_\Pi \|z - \Pi\widehat{z}\|_\infty/\|z\|_\infty$, where we minimize over all six permutations P.

First, we arbitrarily take the "+" square root in (24.5). With almost any starting vector of coefficients, the MDS maximizer rapidly identifies coefficients for which the computed roots are very inaccurate. For example, starting with $[1, 1, 1]^T$ we are lead to the vector

$$[a \;\; b \;\; c]^T = [1.732 \;\; 1 \;\; 1.2704]^T,$$

for which the computed and "exact" roots are

$$\widehat{z} = \begin{bmatrix} -1.5999\text{e+0} \\ -6.6066\text{e-2} - 8.8557\text{e-1}i \\ -6.6066\text{e-2} + 8.8557\text{e-1}i \end{bmatrix}, \qquad z = \begin{bmatrix} -1.6026\text{e+0} \\ -6.4678\text{e-2} + 8.8798\text{e-1}i \\ -6.4678\text{e-2} - 8.8798\text{e-1}i \end{bmatrix}.$$

---

[21] Edelman and Murakami [346, 1995] and Toh and Trefethen [1007, 1994] analyse the stability of this method of finding polynomial roots; the method is stable.

When the cubic is evaluated at the computed roots $\widehat{z}$ the results are of order 10^{-2}, whereas they are of order 10^{-15} for z. Since the roots are well separated the problem of computing them is not ill conditioned, so we can be sure that z is an accurate approximation to the exact roots. The conclusion is that the formulae, as programmed, are numerically unstable.

Recalling the discussion for a quadratic equation in §1.8, a likely reason for the observed instability is cancellation in (24.5). Instead of always taking the "+" sign, we therefore take

$$w^3 = -\frac{q}{2} - \operatorname{sign}(q)\sqrt{\frac{q^2}{4} + \frac{p^3}{27}}. \tag{24.6}$$

When the argument of the square root is nonnegative, this formula suffers no cancellation in the subtraction; the same is true when the argument of the square root is negative, because then the square root is pure imaginary. With the use of (24.6), we were unable to locate instability using the objective function described above. However, an alternative objective function can be derived as follows. It is reasonable to ask that the computed roots be the exact roots of a slightly perturbed cubic. Thus each computed root $\widehat{z}_i$ should be a root of

$$(1+\Delta d)x^3 + (a+\Delta a)x^2 + (b+\Delta b)x + c + \Delta c = 0,$$

where $\max(|\Delta a|, |\Delta b|, |\Delta c|, |\Delta d|)/\max(|a|, |b|, |c|, 1)$ is of the order of the unit roundoff. Notice that we are allowing the leading coefficient of unity to be perturbed. Denoting the unperturbed cubic by f, we find that this condition implies that

$$\max_i \frac{|f(\widehat{z}_i)|}{\max(|a|, |b|, |c|, 1)\sum_{j=0}^{3}|\widehat{z}_i^j|} \tag{24.7}$$

is of order u. We therefore take the quantity in (24.7) as the function to be maximized. On applying the MDS maximizer with starting vector $[1, 1, 1]^T$, we obtain after 10 iterations an objective function value of 1.2×10^{-11}. The cubic coefficient vector is (to three significant figures)

$$[a \quad b \quad c]^T = [-5.89 \times 10^2 \quad 3.15 \times 10^2 \quad -1.36 \times 10^1],$$

and the computed roots are (ignoring tiny, spurious imaginary parts)

$$\widehat{z}^T = [4.75 \times 10^{-2} \quad 4.87 \times 10^{-1} \quad 5.89 \times 10^2].$$

The value of the objective function corresponding to the "exact" roots (computed as described above) is of order 10^{-16} (and the value of the previous relative error objective function for the computed roots is of order 10^{-14}).

The conclusion is that even using (24.6) the formulae for the roots are numerically unstable. However, further theoretical analysis is needed to understand the stability fully; see Problem 24.3.

24.4. Interval Analysis

Interval analysis has been an area of extensive research since the 1960s, and it had been considered earlier by Turing and Wilkinson in the 1940s [1099, 1980, p. 104]. As the name suggests, the idea is to perform arithmetic on intervals $[a,b]$ $(b \geq a)$. The aim is to provide as the solution to a problem an interval in which the desired result is guaranteed to lie, which may be particularly appropriate if the initial data is uncertain (and hence can be thought of as an interval).

For the elementary operations, arithmetic on intervals is defined by

$$[a,b] \text{ op } [c,d] = \{\, x \text{ op } y : x \in [a,b],\ y \in [c,d] \,\},$$

and the results are given directly by the formulae

$$\begin{aligned}
[a,b] + [c,d] &= [a + c,\ b + d],\\
[a,b] - [c,d] &= [a - d,\ b - c],\\
[a,b] * [c,d] &= [\min(ac, ab, bc, bd),\ \max(ac, ab, bc, bd)],\\
[a,b] / [c,d] &= [a,b] * [1/d,\ 1/c], \qquad 0 \notin [c,d].
\end{aligned}$$

We will use the notation $[x]$ for an interval $[x_l,x_2]$ and we define width(x) $:= x_2 - x_1$.

In floating point arithmetic, an interval containing $fl([x] \text{ op } [y])$ is obtained by rounding computations on the left endpoint to $-\infty$ and those on the right endpoint to $+\infty$ (both these rounding modes are supported in IEEE arithmetic).

The success of an interval analysis depends on whether an answer is produced at all (an interval algorithm will break down with division by zero if it attempts to divide by an interval containing zero), and on the width of the interval answer. A one-line program that prints "$[-\infty,\infty]$" would be correct, robust, and fast, but useless. Interval analysis is controversial because, in general, narrow intervals cannot be guaranteed. One reason is that when dependencies occur in a calculation, in the sense that a variable appears more than once, final interval lengths can be pessimistic. For example, if $[x] = [1,2]$ then

$$[x] - [x] = [-1,\ 1], \quad [x]/[x] = [1/2,\ 2],$$

whereas the optimal intervals for these calculations are, respectively, [0, O] and [1, 1]. These calculations can be interpreted as saying that there is no additive or multiplicative inverse in interval arithmetic.

An example of an algorithm for which interval arithmetic can be ineffective is GEPP, which in general gives very pessimistic error bounds and is unstable in interval arithmetic even for a well-conditioned matrix [798, 1977]. The basic problem is that the interval sizes grow exponentially. For example, in the 2 x 2 reduction

$$\begin{bmatrix} 1 & [x] \\ 1 & [y] \end{bmatrix} \rightarrow \begin{bmatrix} 1 & [x] \\ 0 & [y]-[x] \end{bmatrix},$$

if $[x] \approx [y]$ then

$$\mathrm{width}([y]-[x]) \approx \mathrm{width}([x]-[x]) = 2\,\mathrm{width}([x]).$$

This type of growth is very likely to happen, unlike the superficially similar phenomenon of element growth in standard GEPP. The poor interval bounds are entirely analogous to the pessimistic results returned by early forward error analyses of GEPP (see §9.6). Nickel [798, 1977] states that “The interval Gauss elimination method is one of the most unfavorable cases of interval computation . . . nearly all other numerical methods give much better results if transformed to interval methods”. Interval GE is effective, however, for certain special classes of matrices, such as M-matrices.

As already mentioned, there is a large body of literature on interval arithmetic, though, as Skeel notes (in an article that advocates interval arithmetic), “elaborate formalisms and abstruse notation make much of the literature impenetrable to all but the most determined outsiders” [922, 1989]. Good sources of information include various conference proceedings and the journal *Computing*. The earliest book is by Moore [778, 1966], whose later book [779, 1979] is one of the best references on the subject. Nickel [798, 1977] gives an easy to read summary of research up to the mid 1970s. A more recent reference is Alefeld and Herzberger [10, 1983].

Yohe [1120, 1979] describes a Fortran 66 package for performing interval arithmetic in which machine-specific features are confined to a few modules. It is designed to work with a precompiled for Fortran called Augment [254, 1979], which allows the user to write programs as though Fortran had extended data types—in this case an INTERVAL type. A version of the package that allows arbitrary precision interval arithmetic by incorporating Brent’s multiple precision arithmetic package (see §25.9) is described in [1121, 1980]. In [1119, 1979], Yohe describes general principles of implementing nonstandard arithmetic in software, with particular reference to interval arithmetic.

Kulisch and Miranker have proposed endowing a computer arithmetic with a super-accurate inner product, that is, a facility to compute an exactly rounded inner product for any two vectors of numbers at the working precision [678, 1981], [679, 1983], [680, 1986]. This idea has been implemented in the package ACRITH from IBM, which employs interval arithmetic [124, 1985]. Anomalies in early versions of ACRITH are described by Kahan and LeBlanc [639, 1985] and Jansen and Weidner [612, 1986]. For a readable discussion of interval arithmetic and the super-accurate inner product, see Rail [858, 1991]. Cogent arguments against adopting a super-accurate inner product are given by Demmel [284, 1991].

Fortran and Pascal compilers (Fortran SC and Pascal-XSC) and a C++ class library that support a super-accurate inner product and interval arithmetic have been developed jointly by researchers at IBM and the University of Karlsruhe [125, 1987], [661, 1993], [662, 1992]. A toolbox of routines written in Pascal-XSC for solving basic numerical problems and verifying the results is presented by Hammer, Hocks, Kulisch, and Ratz [500, 1993].

Finally, we explain why any attempt to compute highly accurate answers using interval arithmetic of a fixed precision (possibly combined with a super-accurate inner product) cannot always succeed. Suppose we have a sequence of problems to solve, where the output of one is the input to the next: $x_{i+1} = f_i(x_i)$, $i = 1{:}n$, for smooth functions $f_i : \mathbb{R} \rightarrow \mathbb{R}$. Suppose x_1 is known exactly (interval width zero). Working in finite precision interval arithmetic, we obtain an interval $[x_2 - \alpha_2, x_2 + b_2]$ containing x_2. This is used as input to the computation of f_2. Even under the favorable assumption of no rounding errors in the evaluation of f_2, we obtain an interval answer whose width must be of order

$$|f_2(x_2) - f_2(x_2 \pm \epsilon_2)| \gtrsim |f_2'(x_2)|\epsilon_2, \qquad \epsilon_2 = \min(|\alpha_2|, |\beta_2|)$$

(the interval could be much bigger than $f_2(x_2) - f_2(x_2 \pm e_2)$, depending on the algorithm used to evaluate f_2). In other words, the width of the interval containing x_3 is roughly proportional to the condition number of f_2. When the output of the f_2 computation is fed into f_3 the interval width is multiplied by f'_3 The width of the final interval containing x_{n+1} is proportional to the product of the condition numbers of all the functions f_i and if there are enough functions, or if they are conditioned badly enough, the final interval will provide no useful information The only way to avoid such a failure is to use variable precision arithmetic or to reformulate the problem to escape the product of the condition numbers of the f_i.

24.5. Other Work

In this section we outline other work on automatic error analysis.

In the 1970s, Miller and his co-authors developed methods and software for automatically searching for numerical instability in algebraic processes [757, 1975], [760, 1978], [762, 1980]. In [757, 1975] Miller defines a quantity $s(d)$ that bounds, to first order, the sensitivity of an algorithm to perturbations in the data d and in the intermediate quantities that the algorithm generates. He then defines the forward stability measure $r(d) = s(d)/_k(d)$, where $_k(d)$ is a condition number for the problem under consideration. The algorithms to be analysed are required to contain no loops or conditional branches and are presented to Miller's Fortran software in a special numeric encoding. The software automatically computes the partial derivatives needed to evaluate

$r(d)$, and attempts to maximize r using the method of alternating directions. Miller gives several examples illustrating the scope of his software; he shows, for example, that it can identify the instability of the classical Gram-Schmidt method for orthogonalizing a set of vectors.

In [760, 1978], [761, 1978] Miller and Spooner extend the work in [757, 1975] in several ways. The algorithm to be analysed is expressed in a Fortran-like language that allows for-loops but not logical tests. The definition of p is generalized and a method of computing it is developed that involves solving a generalized eigenvalue problem. The book by Miller and Wrathall [762, 1980] gives a thorough development of the work of [760, 1978], including a description of the graph theory techniques used to compute the partial derivatives, and it provides further examples of the use of the software. The potential of Miller and Spooner's software for exposing numerical instability is clearly demonstrated by the case studies in these references, yet the software has apparently not been widely used. This is probably largely due to the inability of the software to analyse algorithms expressed in Fortran, or any other standard language.

A different approach to algorithm analysis is taken by Larson and Sameh [688, 1978], [689, 1980], and implemented in software by Larson, Pasternak, and Wisniewski [687, 1983]. Here, errors are measured in a relative rather than an absolute sense, and the stability is analysed at *fixed data* instead of attempting to maximize instability over all data; however, the analysis is still linearized.

The idea of applying automatic differentiation to a computational graph to obtain estimates for the forward error in an algorithm is found not only in the references cited above, but also in the automatic differentiation literature; see Rail [857, 1981], Iri [606, 1991], and Kubota [675, 1991], for example.

Hull [589, 1979] discusses the possibility of applying program verification techniques to computations that are subject to rounding errors, with the aim of proving a program "correct". Difficulties include deciding what "correct" means and formulating appropriate assertions. Although he reports some progress, Hull concludes that "there is not a great deal to be gained by trying to apply the techniques of program verification to programs for numerical calculations".

Bliss, Brunet, and Gallopoulos [126, 1992] develop Fortran preprocessor tools for implementing the local relative error approach of Larson and Sameh. Their tools also implement a statistical technique of Brunet and Chatelin [152, 1989], [202, 1990] in which the result of every floating point operation is randomly perturbed and statistical methods are used to measure the effect on the output of the algorithm.

Rowan [882, 1990] develops another way to search for numerical instability. For an algorithm with data d he maximizes $S(d) = e(d)/_{k}(d)$ using a direct search maximizer he has developed called the subplex method (which is based on the Nelder–Mead simplex method). Here, $e(d) = y_{acc} - \widehat{y}$ is an approximation to the forward error in the computed solution $\widehat{y}$, where y_{acc} is a more accurate estimate of the true solution than $\widehat{y}$, and the condition number $\kappa(d)$ is estimated using finite difference approximations. The quantity $S(d)$ is a lower bound on the backward error of the algorithm at d. Fortran software given in [882, 1990] implements this "functional stability analysis". The software takes as input two user-supplied Fortran subprograms; one implements the algorithm to be tested in single precision, and the other provides a more accurate solution, typically by executing the same algorithm in double precision. The examples in [882, 1990] show that Rowan's software is capable of detecting numerical instability in a wide variety of numerical algorithms.

A technique called "significance arithmetic" was studied by Ashenhurst, Metropolis, and others in the 1960s. It involves performing arithmetic on unnormalized numbers in such a way that the number of significant digits in the answers provides estimates of the accuracy of those answers. Significance arithmetic is therefore a form of automatic error analysis. For more details see, for example, Ashenhurst and Metropolis [31, 1965], [750, 1977] and Sterbenz [938, 1974, §7.2]; Sterbenz explains several drawbacks of the technique.

Stoutemyer [958, 1977] describes the use of the symbolic manipulation package REDUCE to analyse the propagation of data and rounding errors in numerical algorithms.

Finally, we note that running error analysis (see §3.3) is a form of automatic error analysis and is an attractive alternative to interval arithmetic as a means of computing a posteriori error bounds for almost any numerical algorithm.

24.6. Notes and References

Sections 24.1 – 24.3.2 and §24.5 are based on Higham [555, 1993].

The Release Notes for MATLAB 4.1 state that "This release of MATLAB fixes a bug in the `rcond` function. Previously, `rcond` returned a larger than expected estimate for some matrices . . . `rcond` now returns an estimate that matches the value returned by the Fortran LINPACK library." In the direct search experiments in [555, 1993] we used MATLAB 3.5, and we found it much easier to generate counterexamples to `rcond` than we do now with MATLAB 4.2. It seems that the maximizations in [555, 1993] were not only defeating the algorithm underlying `rcond`, but also, unbeknown to us, exploiting a bug in the implementation of the function. The conclusions of [555, 1993] are unaffected, however.

Another way to solve a cubic is to use Newton's method to find a real zero, and then to find the other two zeros by solving a quadratic equation.

In a detailed investigation, Kahan [631, 1986] finds this iterative method **to** be preferable to (sophisticated) use of the explicit formulae. Other useful references on the numerical computation of roots of a cubic are Lanczos [686, 1956, Chap. 1], Press, Teukolsky, Vetterling, and Flannery [842, 1992, §5.6], and Uspensky [1038, 1948].

Problems

24.1. Let $A \in \mathbb{R}^{4\times 4}$ and let $\widehat{Q}_{CGS}$ and $\widehat{Q}_{MGS}$ be the computed orthogonal QR factors produced by the classical and modified Gram-Schmidt methods, respectively. Use direct search **to** maximize the functions

$$\begin{aligned} f_1(A) &= \|\widehat{Q}_{CGS}^T\widehat{Q}_{CGS} - I\|_2, \\ f_2(A) &= \|\widehat{Q}_{MGS}^T\widehat{Q}_{MGS} - I\|_2, \\ f_3(A) &= f_1(A)/f_2(A). \end{aligned}$$

In order **to** keep $k_2(A)$ small, try maximizing $f_i(A) - q\max(k_2(A) - m, 0)$, where q is a large positive constant and m is an upper bound on the acceptable condition numbers.

24.2. It is well known that if $A \in \mathbb{R}^{n\times n}$ is nonsingular and $v^TA^{-1}u \neq -1$ then

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}.$$

This is known as the Sherman-Morrison formula. For a history and generalizations see Henderson and Searle [512, 1981]. A natural question is whether this formula provides a stable way to solve a rank-1 perturbed linear system. That is, is the following algorithm stable?

```
% Solve $Cx := (A + uv^T)x = b$.
Solve $Ay = b$ for $y$.
Solve $Az = u$ for $z$.
$x = y - (v^Ty)(1 + v^Tz)^{-1}z$
```

(a) Investigate the stability using direct search. Let both linear systems with coefficient matrix A be solved by GEPP. Take A, u, and v as the data and let the function to be maximized be the normwise backward error $h_{C,b}$ in the ∞-norm.

(b) (RESEARCH PROBLEM) Obtain backward and forward error bounds for the method (for some existing analysis see Yip [1118, 1986]).

24.3. (RESEARCH PROBLEM) Investigate the stability of the formulae of $24.3.3 for computing the roots of a cubic.

Chapter 25

Software Issues in Floating Point Arithmetic

The first American Venus probe was lost due to a
program fault caused by the
inadvertent substitution of a statement of the form
`DO 3 I = 1.3` *for one of the form* `DO 3 I = 1,3.`
— JIM HORNING, Note on Program Reliability[22] (1979)

Numerical subroutines should deliver results that satisfy simple,
useful mathematical laws whenever possible.
— DONALD E. KNUTH, *The Art of Computer*
Programming, Volume 2, Seminumerical Algorithms (1981)

No method of solving a computational problem is
really available to a user until it is
completely *described in an algebraic computing language*
and made completely reliable.
Before that, there are indeterminate aspects in any algorithm.
— GEORGE E. FORSYTHE,
Today's Computational Methods of Linear Algebra (1967)

The extended precision calculation of pi has substantial application as a
test of the "global integrity" of a supercomputer . . .
Such calculations . . . are apparently now used routinely
to check supercomputers before they leave the factoty.
A large-scale calculation of pi is entirely unforgiving;
it soaks into all parts of the machine and a
single bit awry leaves detectable consequences.
— J. M. BORWEIN, P. B. BORWEIN, and D. H. BAILEY,
Ramanujan, Modular Equations, and Approximations to Pi
or How to Compute One Billion Digits of Pi (1989)

---

[22] Quoted, with further details, in Tropp [1021, 1984].

In this chapter we discuss some miscellaneous aspects of floating point arithmetic that have an impact on software development.

25.1. Exploiting IEEE Arithmetic

IEEE standard 754 and 854 arithmetic can be exploited in several ways in numerical software, provided that proper access to the arithmetic's features is available in the programming environment. Unfortunately, although most commercially significant floating point processors at least nearly conform to the IEEE standards, language standards and compilers generally provide poor support (exceptions include the Standard Apple Numerics Environment (SANE) [233, 1988], Apple's PowerPC numerics environment [234, 1994], and Sun's SPARCstation compilers [969, 1992], [970, 1992]). We give four examples to show the benefits of IEEE arithmetic on software.

Suppose we wish to evaluate the dual of the vector p-norm $(1 \le p \le \infty)$, that is, the g-norm, where $p^{-1} + q^{-1} = 1$. In MATLAB notation we simply write `norm (x, 1/(1-l/p)),` and the extreme cases $p = 1$ and ∞ correctly yield $q = 1/(1 - 1/p) = \infty$ and 1 in IEEE arithmetic. (Note that the formula $q = p/(p - 1)$ would not work, because inf/inf evaluates to a NaN.)

The following code finds the biggest element in an array $a(1: n)$, and elegantly solves the problem of choosing a "sufficiently negative number" with which to initialize the variable max:

```
max = –inf
for $j$ = 1:$n$
    if $a_j$ > max
        max = $aj$
    end
end
```

Any unknown or missing elements of the array a could be assigned NaNs `(a(j) = NaN)` and the code would (presumably) still work, since a NaN compares as unordered with everything.

Consider the following continued fraction [630, 1981]:

$$r(x) = 7 - \cfrac{3}{x - 2 - \cfrac{1}{x - 7 + \cfrac{10}{x - 2 \cfrac{2}{x - 3}}}},$$

which is plotted over the range [0,4] in Figure 25.1. Another way to write the

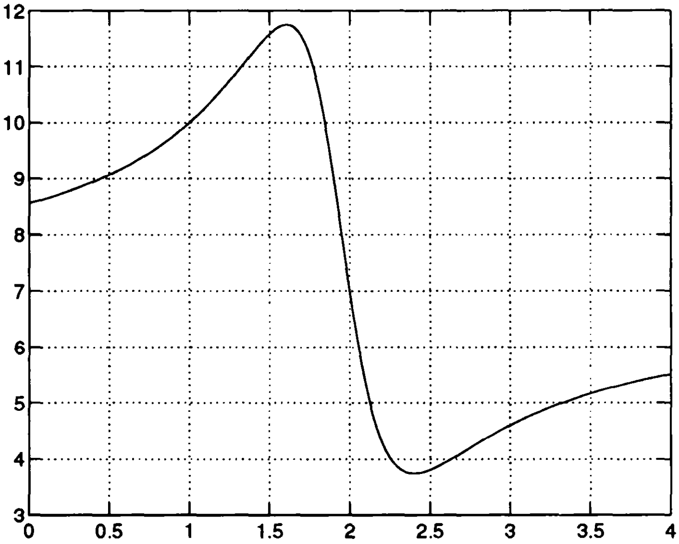


Figure 25.1. *Rational function r.*

rational function $r(x)$ is in the form

$$r(x)=\frac{(((7x-101)x+540)x-1204)x+958}{(((x-14)x+72)x-151)x+112},$$

in which the polynomials in the numerator and denominator are written in the form in which they would be evaluated by Homer's rule. Examining these two representations of r, we see that the continued fraction requires less arithmetic operations to evaluate (assuming it is evaluated in the obvious "bottom up" fashion) but it incurs division by zero at the points $x = 1:4$, even though r is well behaved at these points. However, in IEEE arithmetic r evaluates correctly at these points because of the rules of infinity arithmetic. For $x = 10^{77}$, the rational form suffers overflow, while the continued fraction evaluates correctly to 7.0; indeed, in IEEE arithmetic the continued fraction is immune to overflow. Figure 25.2 shows the relative errors made in evaluating r *in* double precision on an equally spaced grid of points on the range [0,4] (many of the errors for the continued fraction are zero, hence the gaps in the error curve); clearly, the continued fraction produces the more accurate function values. The conclusion is that in IEEE arithmetic the continued fraction representation is the preferred one for evaluating r.

The exception handling provided by IEEE arithmetic can be used to simplify and speed up certain types of software, provided that the programming environment properly supports these facilities. Recall that the exceptional

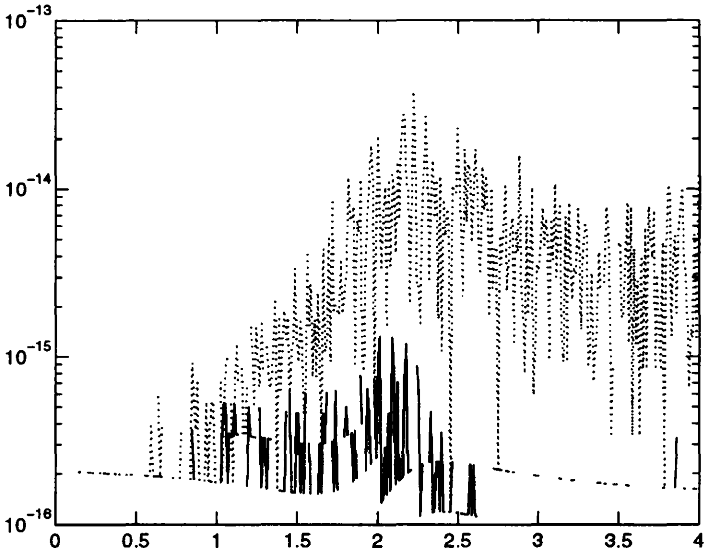


Figure 25.2. *Error in evaluating rational function r. Solid line: continued fraction, dotted line: usual rational form.*

operations underflow, overflow, divide by zero, invalid operation, and inexact deliver a result, set a flag, and continue. The flags (one for each type of exception) are “sticky”, remaining set until explicitly cleared, and implementations of the IEEE standard are required to provide a means to read and write the flags. Unfortunately, compiler writers have been slow to make exception handling, and some of IEEE arithmetic’s other unusual features, available to the programmer. Kahan has commented that “the fault lies partly with the IEEE standard, which neglected to spell out standard names for the exceptions, their flags, or even ∞.”

As an example of how exception handling can be exploited, consider the use of the LAPACK norm estimator (Algorithm 14.4) to estimate $\|A^{-1}\|_1$. The algorithm requires the solution of linear systems of the form $Ax = b$ and $A^Ty = c$, which is typically done with the aid of an LU factorization $PA = LU$. Solution of the resulting triangular systems is prone to both overflow and division by zero. In LAPACK, triangular systems are solved in this context not with the level-2 BLAS routine xTRSV but with a routine xLATRS that contains elaborate tests and scalings to avoid overflow and division by zero [18, 1991]. In an IEEE arithmetic environment a simpler and potentially faster approach is possible: solve the triangular systems with xTRSV and after computing each solution check the flags to see whether any exceptions occurred. Provided that some tests are added to take into account overflow due solely to $\|A\|_1$ being tiny, the occurrence of an exception allows the conclusion that A is singular to working precision; see Demmel and Li [298, 1994] for the details.

For the condition estimator that uses exception handling to be uniformly faster than the code with scaling it is necessary that arithmetic with NaNs, infinities , and subnormal numbers be performed at the same speed as conventional arithmetic. While this requirement is often satisfied, there are machines for which arithmetic on NaNs, infinities, and subnormal numbers is between one and three orders of magnitude slower than conventional arithmetic [298, 1994, Table 2]. Demmel and Li compared the LAPACK condition estimation routines `xxxCON` with modified versions containing exception handling and found the latter to be up to 6 times faster and up to 13 times slower, depending on the machine and the matrix.

Appendices to the IEEE standards 754 and 854 recommend 10 auxiliary functions and predicates to support portability of programs across different IEEE systems. These include nextafter(x,y), which returns the next floating point number to x in the direction of y, and scalb(x,n), which returns $x \times b^n$ without explicitly computing b^n, where n is an integer and b the base. Not all implementations of IEEE arithmetic support these functions. Portable C versions of six of the functions are presented by Cody and Coonen [226, 1993].

25.2. Subtleties of Floating Point Arithmetic

The difference $x - y$ of two machine numbers can evaluate to zero even when $x \neq y$, because of underflow. This makes a test such as

```
if $x \neq y$
  $f = f/(x - y)$
end
```

unreliable. However, in a system that supports gradual underflow (such as IEEE arithmetic) $x - y$ always evaluates as nonzero when $x \neq y$, as is easy to verify. On several models of Cray computer (Cray 1, 2, X-MP, Y-MP, and C90) this code could fail for another reason: they compute $f/(x - y)$ as $f * (1/(x - y))$ and $1/(x - y)$ could overflow.

It is a general principle that one should not test two floating point numbers for equality, but rather use a test such as if $|x - y| \leq$ tol (there are exceptions, such as Algorithm 2 in §1.14.1). Of course, skill may be required to choose an appropriate value for tel. A test of this form would correctly avoid the division in the example above when underflow occurs.

Table 25.1. *Results from Cholesky factorization.*

Bits	u	Computer	Displacement
128	le-29	Cray 2	.447440341
64	le-17	IBM 3090	.44744034*4*
64	le-16	Convex 220	.44744033*9*
64	le-16	IRIS	.44744033*9*
64	le-15	Cray 2	.4474403*03*
64	le-15	Cray Y-MP	.44743*6106*

25.3. Cray Peculiarities

Carter [188, 1991] describes how a Cray computer at the NASA Ames Laboratories produced puzzling results that were eventually traced to properties of its floating point arithmetic. Carter used Cholesky factorization on a Cray Y-MP to solve a sparse symmetric positive definite system of order 16146 resulting from a finite element model of the National Aerospace Plane. The results obtained for several computers are shown in Table 25.1, where "Displacement" denotes the largest component of the solution and incorrect digits are set in italics and underlined. Since the coefficient matrix has a condition number of order 10^{11}, the errors in the displacements are consistent with the error bounds for Cholesky factorization.

Given that both the Cray 2 and the Cray Y-MP lack guard digits, it is not surprising that they give a less accurate answer than the other machines with a similar unit roundoff. What is surprising is that, even though both machines use 64-bit binary arithmetic with a 48-bit mantissa, the result from the Cray Y-MP is much less accurate than that from the Cray 2. The explanation (diagnosed over the telephone to Carter by Kahn, as he scratched on the back of an envelope) lies in the fact that the Cray 2 implementation of subtraction without a guard digit produces more nearly unbiased results (average error zero), while the Cray Y-MP implementation produces biased results, causing $fl(x-y)$ to be too big if $x > y > 0$. Since the inner loop of the Cholesky algorithm contains an operation of the form $a_{ii} = a_{ii} - a_{ij}^2$, the errors in the diagonal elements reinforce as the factorization proceeds on the Cray Y-MP, producing a Cholesky factor with a diagonal that is systematically too large. Similarly, since Carter's matrix has off-diagonal elements that are mostly negative or zero, the Cray Y-MP produces a Cholesky factor with off-diagonal elements that are systematically too large and negative. For the large value of n used by Carter, these two effects in concert are large enough to cause the loss of a further two digits in the answer.

An inexpensive way to improve the stability of the computed solution is to use iterative refinement in fixed precision (see Chapter 11). This was tried by Carter. He found that after one step of refinement the Cray Y-MP solution was almost as accurate as that from the Cray 2 without refinement.

25.4. Compilers

Some of the pitfalls a compiler writer should avoid when implementing floating point arithmetic are discussed by Farnum [364, 1988]. The theme of his paper is that programmers must be able to predict the floating point operations that will be performed when their codes are compiled; this may require, for example, that the compiler writer forgoes certain "optimizations". In particular, compilers should not change groupings specified by parentheses. For example, the two expressions

```
(1.0E-30 + 1.0E+30) - 1.0E+30
 1.0E-30 + (1.0E+30 - 1.0E+30)
```

will produce different answers on many machines. Farnum explains that

> Compiler texts and classes rarely address the peculiar problems of floating-point computation, and research literature on the topic is generally confined to journals read by numerical analysts, not compiler writers. Many production-quality compilers that are excellent in other respects make basic mistakes in their compilation of floating-point, resulting in programs **that** produce patently absurd results, or worse, reasonable but inaccurate results.

25.5. Determining Properties of Floating Point Arithmetic

Clever algorithms have been devised that attempt to reveal the properties and parameters of a floating point arithmetic. The first algorithms were published by Malcolm [724, 1972] (see Problem 25.3). These have been improved and extended by other authors. Kahan's `paranoia` code carries out detailed investigations of a computer's floating point arithmetic; there are Basic, C, Modula, Pascal, and Fortran versions, all available from netlib. In addition to computing the arithmetic parameters, `paranoia` tries to determine how well the arithmetic has been implemented (so it can be regarded as a program to test a floating point arithmetic—see the next section). Karpinski [645, 1985] gives an introduction to `paranoia` for the layman, but the best documentation for the code is the output it produces.

Cody has a Fortran routine `macher` for determining 13 parameters associated with a floating point arithmetic system [222, 1988] (an earlier version was published in the book by Cody and Waite [228, 1980]). Cody notes some strange behaviour of compilers and says that he was unable to make his code run correctly on the Cyber 205. A routine based on `machar` is given in *Numerical Recipes* [842, 1992, §20.1].

LAPACK contains a routine `xLAMCH` for determining machine parameters. Because of the difficulty of handling the existing variety of machine arithmetics it is over 850 lines long (including comments and the subprograms it calls). Programs such as `machar`, paranoia, and `xLAMCH` are difficult to write; for example, `xLAMCH` tries to determine the overflow threshold without invoking overflow. The Fortran version of `paranoia` handles overflow by requiring the user to restart the program, after which checkpointing information previously written to a file is read to determine how to continue.

Fortran 90 contains environmental inquiry functions, which for `REAL` arguments return the precision (`PRECISION`[23]), exponent range (`RANGE`), machine epsilon (`EPSILON`), largest number (`HUGE`), and so on, corresponding to that argument [749, 1990]. The values of these parameters are chosen by the implementor to best fit a model of the machine arithmetic due to Brown [150, 1981] (see §25.7.4). Fortran 90 also contains functions for manipulating floating point numbers: for example, to set or return the exponent or fractional part (`EXPONENT`, `FRACTION`, `SET_EXPONENT`) and to return the spacing of the numbers having the same exponent as the argument (`SPACING`).

25.6. Testing a Floating Point Arithmetic

How can we test whether a particular implementation of floating point arithmetic is correct? It is impractical to test a floating point operation with all possible arguments because there are so many of them—about 10^{19} in IEEE double precision arithmetic, for example! Special techniques are therefore needed that test with a limited number of arguments.

A package FPV [784, 1986] from NAG Ltd. attempts to verify experimentally that a floating point arithmetic has been correctly implemented according to its specification. FPV must be supplied by the user with the arithmetic parameters b, t, $e_{\min}$, $e_{\max}$, and with the rounding rule; it then attempts to verify that the arithmetic conforms to these parameters by probing for errors. The tests cover the basic arithmetic operations (including square root) but not the elementary functions. FPV, which is available in both Fortran 77 and Pascal versions, adopts an approach used in an earlier program FPTST [906, 1981], [907, 1984] by Schryer of AT&T Bell Laboratories: it tests arithmetic operations with a limited set of operands that are regarded as being most likely to reveal errors. This approach is based on the premise that errors are most likely to occur as "edge effects", induced by operands close to a discontinuity or boundary of the floating point number system (such as a power of the base b).

---

[23] Intrinsic function names are shown in parentheses.

FPV and FPTST have both revealed implementation errors in floating point arithmetics on commercial computers. Errors detected include multiplication and negation producing unnormalized results, $x * y$ differing from $(-x) * (-y)$, and the product of two numbers greater than 1 being less than 1! Wichmann [1078, 1992] suggests that it was probably revelations such as these that led the UK Ministry of Defence to issue an interim standard prohibiting the use of floating point arithmetic in safety-critical systems.

A Fortran package ELEFUNT by Cody and Waite contains programs to test the elementary functions [221, 1982], [228, 1980]; the package is available from netlib. It checks identities such as $\cos(x) = \cos(x/3)(4\cos^2(x/3) - 1)$, taking care to choose arguments x for which the errors in evaluating the identities are negligible. A package CELEFUNT serves the same purpose for the complex elementary functions [224, 1993]. Tang [988, 1990] develops table-driven techniques for testing the functions exp and log.

25.7. Portability

Software is portable if it can be made run on different systems with just a few straightforward changes (ideally, we would like to have to make no changes, but this level of portability is often impossible to achieve). Sometimes the word "transportable" is used to describe software that requires certain well-defined changes to run on different machines. For example, Demmel, Dongarra, and Kahan [290, 1992] describe LAPACK as "a transportable way to achieve high efficiency on diverse modern machines", noting that to achieve high efficiency the BLAS need to be optimized for each machine. A good example of a portable collection of Fortran codes is LINPACK. It contains no machine-dependent constants and uses the PFORT subset of Fortran 77; it uses the level-1 BLAS, so, ideally, optimized BLAS would be used on each machine.

25.7.1. Arithmetic Parameters

Differences between floating point arithmetics cause portability problems. First, what is meant by REAL and DOUBLE PRECISION in Fortran varies greatly between machines, as shown by the following table:

	u (REAL)	u (DOUBLE PRECISION)	u (EXTENDED)
IBM	$\approx 10^{-7}$	$\approx 10^{-16}$	$\approx 10^{-31}$
Cray	$\approx 10^{-15}$	$\approx 10^{-28}$	

Second, for a given level of precision, u, the various arithmetic parameters such as base, unit roundoff, largest and smallest machine numbers, and the type of rounding, can all vary. Some possible ways for a code to acquire these parameters are as follows.

(1) The parameters are evaluated and embedded in the program in `PARAMETER` and `DATA` statements. This is conceptually the simplest approach, but it is not portable.

(2) Functions are provided that return the machine parameters. Bell Laboratories' PORT library [405, 1978] has three functions:

`R1MACH(k)`	$1 \leq k \leq 5$	real parameters,
`D1MACH(k)`	$1 \leq k \leq 5$	double precision parameters,
`I1MACH(k)`	$1 \leq k \leq 16$	integer parameters.

`R1MACH` returns the underflow and overflow thresholds, the smallest and largest relative spacing (b^{-t}, b^{1-t} respectively), and $\log_{10} b$, where b is the base and t the number of digits in the mantissa. `I1MACH` returns standard input, output and error units and further floating point information, such as b, t, and the minimum and maximum exponents $e_{\min}$ and $e_{\max}$. These functions do not carry out any computation; they contain `DATA` statements with the parameters for most common machines in comment lines, and the relevant statements have to be uncommented for a particular machine. This approach is more sensible for a library than the previous one, because only these three routines have to be edited, instead of every routine in the library.

The NAG Library takes a similar approach to PORT. Each of the 18 routines in its X02 chapter returns a different machine constant. For example, `XO2AJF` returns the unit roundoff and `X02ALF` returns the largest positive floating point number. These values are determined once and for all when the NAG library is implemented on a particular platform using a routine similar to `paranoia` and `machar`, and they are hard coded into the Chapter X02 routines.

(3) The information is computed at run-time, using algorithms such as those described in §25.5.

25.7.2. 2 x 2 Problems in LAPACK

LAPACK contains 10 or so auxiliary routines for solving linear equation and eigenvalue problems of dimension 2. For example, `SLAS2` computes the singular values of a 2 x 2 triangular matrix, `SLAEV2` computes the eigenvalues and eigenvectors of a 2 x 2 symmetric matrix, and `SLASY2` solves a Sylvester equation $AX - XB = C$ where A and B have order 1 or 2. These routines need to be reliable and robust because they are called repeatedly by higher-level solvers. Because LAPACK makes minimal assumptions about the underlying floating point arithmetic, the 2 x 2 routines are nontrivial to write and are surprisingly long: the counts of executable statements are 39 for `SLAS2`, 61 for

`SLAEV2,` and 207 for `SLASY2.` If the availability of extended precision arithmetic (possibly simulated using the working precision) or IEEE arithmetic can be assumed, the codes can be simplified significantly. Complicated and less efficient code for these 2 x 2 problem solvers is a price to be paid for portability across a wide range of floating point arithmetics.

25.7.3. Numerical Constants

How should numerical constants be specified, e.g., for a Gauss quadrature rule or a Runge–Kutta method? Some compilers limit the number of significant digits that can be specified in a DATA statement or an assignment statement. One possibility is to use rational approximations, but again the number of digits required to cater for all precision may be too large for some compilers. Another scheme is to store a carefully designed table of leading and trailing parts of constants. A constant such as π is best computed via a statement such as `pi = 4.0*atan(1.0),` if you trust your language's mathematics library.

25.7.4. Models of Floating Point Arithmetic

The model (2.4) of floating point arithmetic contains only one parameter, the unit roundoff. Other more detailed models of floating point arithmetic have been proposed, with the intention of promoting the development of portable mathematical software. A program developed under a model and proved correct for that model has the attraction that it necessarily performs correctly on any machine for which the model is valid.

The first detailed model was proposed by van Wijngaarden [1046, 1966]; it contained 32 axioms and was unsuccessful because it was "mathematically intimidating" and did not cover the CDC 600, an important high-performance computer of that time [635, 1991]. A more recent model is that of Brown [150, 1981]. Brown's model contains four parameters: the base b, the precision t, and the minimum and maximum exponents $e_{\min}$ and $e_{\max}$ together with a number of axioms describing the behaviour of the floating point arithmetic. Aberrant arithmetics are accommodated in the model by penalizing their parameters (for example, reducing t by 1 from its actual machine value if there is no guard digit). Brown builds a substantial theory on the model and gives an example program to compute the 2-norm of a vector, which is accompanied by a correctness proof for the model.

Brown's model is intended to cater to diverse computer arithmetics. In fact, "any behaviour permitted by the axioms of the model is actually exhibited by at least one commercially important computer" [150, 1981, p. 457]. This broadness is a weakness. It means that there are important properties shared by many (but not all) machines that the model cannot reflect, such as the exactness of $fl(x - y)$ described in Theorem 2.4. As Kahan [630, 1981] notes, "Programming for the [IEEE] standard is like programming for one of a family of well-known machines, whereas programming for a model is like programming for a horde of obscure and ill-understood machines all at once." Although Brown's model was used in the Ada language to provide a detailed specification of floating point arithmetic, the model is still somewhat controversial.

Wichmann [1077, 1989] gives a formal specification of floating point arithmetic in the VDM specification language based on a modification of Brown's model.

The most recent model is the Language Independent Arithmetic (LIA-1) [702, 1993]. The LIA-1 specifies floating point arithmetic far more tightly than Brown's model. It, too, is controversial; an explanation of flaws in an earlier version (know then as the Language Compatible Arithmetic Standard) was published by Kahan [635, 1991].

For a more detailed critique of models of floating point arithmetic see Priest [844, 1992].

25.8. Avoiding Underflow and Overflow

A classic example showing how care is needed to avoid underflow and overflow is the evaluation of a vector 2-norm, $\|x\|_2 = \sum_i(|x_i|^2)^{1/2}$. For about half of all machine numbers x, x^2 either underflows or overflows! Overflow is avoided by the following algorithm:

```
$t = \|x\|_\infty$
$s = 0$
for $i = 1:n$
    $s = s + (x(i)/t)^2$
end
$\|x\|_2 = t\sqrt{s}$
```

The trouble with this algorithm is that it requires n divisions and two passes over the data (the first to evaluate $\|x\|_\infty$), so it is slow. (It also involves more rounding errors than the unscaled evaluation, which could be obviated by scaling by a power of the machine base.) Blue [128, 1978] develops a one-pass algorithm that avoids both overflow and underflow and requires between O and n divisions, depending on the data, and he gives an error analysis to show that the algorithm produces an accurate answer. The idea behind the algorithm is simple. The sum of squares is collected in three accumulators, one each for small, medium, and large numbers. After the summation, various logical tests are used to decide how to combine the accumulators to obtain the final answer.

The original, portable implementation of the level-l BLAS routine `xNRM2` (listed in [307, 1979]) was written by C. L. Lawson in 1978 and, according to

Lawson, Hanson, Kincaid, and Krogh [694, 1979], makes use of Blue's ideas. However, `xNRM2` is not a direct coding of Blue's algorithm and is extremely difficult to understand—a classic piece of "Fortrans spaghetti"! Nevertheless, the routine clearly works and is reliable, because it has been very widely used without any reported problems. Lawson's version of `xNRM2` has now been superseded in the LAPACK distribution by a concise and elegant version by S. J. Harnmarling, which implements a one-pass algorithm; see Problem 25.5.

A special case of the vector 2-norm is the Pythagorean sum,$\sqrt{a^2+b^2}$, which occurs in many numerical computations. One way to compute it is by a beautiful cubically convergent, square-root-free iteration devised by Moler and Morrison [773, 1983 ; see Problem 25.6. LAPACK has a routine `xLAPY2` that computes $\sqrt{x_1^2+x_2^2}$ and another routine `xLAPY3` that computes $\sqrt{x_1^2+x_2^2+x_3^2}$; both routines avoid overflow by using the algorithm listed at the start of this section with n = 2 or 3.

Pythagorean sums arise in computing the l-norm of a complex vector:

$$\|x\|_1=\sum_{i=1}^{n}\sqrt{(\operatorname{Re}x_i)^2+(\operatorname{Im}x_i)^2},\qquad x\in\mathbb{C}^n.$$

The level-l BLAS routine `xCASUM` does not compute the l-norm, but the more easily computed quantity

$$\sum_{i=1}^{n}(|\operatorname{Re}x_i|+|\operatorname{Im}x_i|).$$

The reason given by the BLAS developers is that it was assumed that users would expect `xCASUM` to compute a less expensive measure of size than the 2-norm [694, 1979]. This reasoning is sound, but many users have been confused not to receive the true l-norm. See Problem 6.16 for more on this pseudo 1-norm.

Another example where underflow and overflow can create problems is in complex division. If we use the formula

$$\frac{a+ib}{c+id}=\frac{ac+bd}{c^2+d^2}+i\frac{bc-ad}{c^2+d^2},$$

then overflow will occur whenever cord exceeds the square root of the overflow threshold, even if the quotient $(a+ib)/(c+id)$ does not overflow. Certain Cray and NEC machines implement complex division in this way [290, 1992]; on these machines the exponent range is effectively halved from the point of view of the writer of robust software. Smith [929, 1962] suggests a simple way to avoid overflow: if $|c|\ge|d|$ use the following formula, obtained by multiplying the numerators and denominators by c^{-1},

$$\frac{a+ib}{c+id}=\frac{a+b(dc^{-1})}{c+d(dc^{-1})}+i\frac{b-a(dc^{-1})}{c+d(dc^{-1})},\tag{25.1}$$

and if $|d| \geq |c|$ use the analogous formula involving d^{-1}. Stewart *[947, 1985]* points out that underflow is still possible in these formulae, and suggests a more complicated algorithm that avoids both underflow and overflow.

Demmel [280, 1984] discusses in detail the effects of underflow on numerical software and analyses different ways of implementing underflows, the main two of which are flush to zero and gradual underflow (as used in IEEE standard arithmetic). Cody [222, 1988] makes the following observations:

> The variety in the treatment of underflow is wondrous. There exist machines in which the exponent wraps around and underflow results in a large floating-point number, perhaps with an accompanying sign change. There are also machines on which underflow is replaced by the smallest in magnitude nonzero number, machines on which small positive numbers vanish when added to themselves, and machines on which such small numbers vanish when multiplied by 1.0. Finally, on IEEE-style machines, underflow may be graceful.

25.9. Multiple Precision Arithmetic

If we wish to work at a precision beyond that provided by a Fortran compiler's `DOUBLE PRECISION` (or the quadruple precision supported as an extension by some compilers) there are two basic approaches. The first is to implement higher precision arithmetic using arrays of double precision variables, where the *unevaluated* sum of the elements of an array represents a high precision variable. Arithmetic can be implemented in this system using techniques such as those of Dekker [275, 1971] (cf. compensated summation, described in §4.3), as refined by Linnainmaa [706, 1981] and Priest [843, 1991], [844, 1992]. These techniques require a "well-behaved" floating point arithmetic, and are not easy to apply.

The second approach is to use arrays to represent numbers in "standard" floating point form with a large base and a long mantissa spread across the elements of an array. All the software described in this section is of the second type.

The first major piece of multiple precision Fortran software was Brent's package [145, 1978], [146, 1978]. This is a portable collection of Fortran 66 subroutines with wide capabilities, including the evaluation of special functions. Brent's package represents multiple precision numbers as arrays of integers and operates on them with integer arithmetic. An interface between Brent's package and the Fortran precompiled Augment [254, 1979] is described by Brent, Hooper, and Yohe [147, 1980]; it makes Brent's package much easier to use.

Recently, Bailey has written a multiple precision arithmetic package MPFUN [44, 1993], which consists of about 10,000 lines of Fortran 77 code in 87 subprograms. In MPFUN, a multiprecision (MP) number is a vector of single precision floating point numbers; it represents a number in base 2^{24} (for IEEE arithmetic). Complex multiprecision numbers are also supported.

MPFUN routines are available to perform the basic arithmetic operations, evaluate *n*th roots and transcendental functions, compare MP numbers, produce a random MP number, solve polynomial equations, and perform other operations. For many of these routines both simple and advanced algorithms are provided; the advanced algorithms are intended for computations at precision levels above 1000 digits. One advanced algorithm is a fast Fourier transform technique for evaluating the convolution involved in a multiplication. Another is used for division: x/y is evaluated as $x * (1/y)$, where $1/y$ is evaluated by Newton's method (see Problem 2.26), which involves only multiplications. An interesting aspect of Newton's method in this context is that it can be implemented with a precision level that doubles at each stage, since the iteration damps out any errors. Another interesting feature of a variable precision environment is that numerical instability can be tackled simply by increasing the precision. As an example, MPFUN does complex multiplication using the 3M method

$$(a+ib)(c+id) = ac - bd + i\big[(a+b)(c+d) - ac - bd\big],$$

which uses three real multiplications instead of the usual four. As we saw in §22.2.4, the 3M method produces a computed imaginary part that can have large relative error, but this instability is not a serious obstacle to its use in MPFUN.

Bailey provides a translator that takes Fortran 77 source containing directives in comments that specify the precision level and which variables are to be treated as multiprecision, and produces a Fortran 77 program containing the appropriate multiprecision subroutine calls. He also provides a Fortran 90 version of the package that employs derived data types and operator extensions [45, 1994]. This Fortran 90 version is a powerful and easy to use tool for doing high precision numerical computations.

Bailey's own use of the packages includes high precision computation of constants such as π [42, 1988] and Euler's constant γ. A practical application to a fluid flow problem is described by Bailey, Krasny, and Pelz [46, 1993]. They found that converting an existing code to use MPFUN routines increased execution times by a factor of about 400 at 56 decimal digit precision, and they comment that this magnitude of increase is fairly typical for multiple precision computation.

Another recent Fortran 77 program for multiple precision arithmetic is the FM package of Smith [926, 1991]. It is functionally quite similar to Bailey's MPFUN. No translator is supplied for converting from standard Fortran 77 source code to code that invokes FM, but Smith notes that the general purpose precompiled Augment [254, 1979] can be used for this purpose.

Fortran routines for high precision computation can also be found in *Numerical Recipes* [842, 1992, §20.6], and high precision numerical computation is supported by many symbolic manipulation packages, including Maple [199, 1991] and Mathematical [1109, 1991].

A Pascal-like programming language called Turing [579, 1988] developed at the University of Toronto in the 1980s is the basis of an extension called Numerical Turing, developed by Hull and his co-workers [591, 1985]. Numerical Turing includes decimal floating point arithmetic whose precision can be dynamically varied throughout the course of a program, a feature argued for in [590, 1982] and implemented in hardware in [231, 1983].

An extension to the level-2 BLAS (see Appendix D) is proposed in [313, 1988, App. B], comprising routines having the same specifications as those in the standard BLAS but which calculate in extended precision.

25.10. Patriot Missile Software Problem

A report from the United States General Accounting Office begins "On February 25, 1991, a Patriot missile defense system operating at Dhahran, Saudi Arabia, during Operation Desert Storm failed to track and intercept an incoming Scud. This Scud subsequently hit an Army barracks, killing 28 Americans" [1036, 1992]. The report finds that the failure to track the Scud missile was caused by a precision problem in the software.

The computer used to control the Patriot missile is baaed on a 1970s design and uses 24-bit arithmetic. The Patriot system tracks its targets by measuring the time it takes for radar pulses to bounce back from them. Time is recorded by the system clock in tenths of a second, but is stored as an integer. To enable tracking calculations the time is converted to a 24 bit floating point number. Rounding errors in the time conversions cause shifts in the system's "range gate", which is used to track the target.

On February 11, 1991 the Patriot Project Office received field data identifying a 20% shift in the Patriot system's range gate after the system had been running continuously for 8 hours. This data implied that after 20 consecutive hours of use the system would fail to track and intercept a Scud. Modified software that compensated for the inaccurate time calculations was released on February 16 by army officials. On February 25, Alpha Battery, which was protecting the Dhahran Air Base, had been in continuous operation for over 100 hours. The inaccurate time calculations caused the range gate to shift so much that the system could not track the incoming Scud. On February 26, the next day, the modified software arrived in Dhahran. Table 25.2, taken from [1036, 1992], shows clearly how, with increasing time of operation, the

Table 25.2. *Effect of extended run time on Patriot missile operation.*

Hours	Seconds	Calculated time (seconds)	Inaccuracy (seconds)	Approximate shift in range gate (meters)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0275	55
20[a]	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100[b]	360000	359999.6567[c]	.3433	687

---

[a] For continuous operation exceeding 20 hours target is outside range gate.
[b] Alpha battery ran continuously for about 100 hours.
[c] Corrected value.

Patriot lost track of its target. Note that the numbers in Table 25.2 are consistent with a relative error of 2^{-20} in the computer's representation of 0.1, this constant being used to convert from the system clock's tenths of a second to seconds (2^{-20} is the relative error introduced by chopping 0.1 to 23 bits after the binary point).

25.11. Notes and References

The issues discussed in this chapter are rarely treated in textbooks. One book that is worth consulting is Miller's *Engineering of Numerical Software* [759, 1984], which is concerned with the designing of reliable numerical software and presents practical examples.

Kahan's paper [627, 1972] is not, as the title suggests, a survey of error analysis, but treats in detail some specific problems, including solution of a quadratic equation and summation, giving error analysis, programs illustrating foibles of Fortran compilers, and ingenious workarounds.

The collection of papers improving *Floating-Point Programming* [1061, 1990] contains, in addition to an introduction to floating point arithmetic and Brown's model, descriptions of recent work on embedding interval arithmetic with a super-accurate inner product in Pascal and ADA (see §24.4).

There are several algorithms for evaluating a continued fraction; see, for example, Blanch [123, 1964] and Press, Teukolsky, Vetterling, and Flannery [842, 1992, §5.2]. Rounding error analysis for the "bottom up" algorithm is given by Jones and Thron [617, 1974].

A standard set of names and definitions of machine parameters was proposed by the IFIP Working Group on Numerical Software (WG 2.5) [383, *1978],* though these do not appear to have been widely adopted.

The use of high precision computations to investigate the Riemann hypothesis and other mathematical problems and conjectures is described by Varga [1052, 1990], who has made use of Brent's package [145, 1978], [146, 1978].

Problems

25.1. (a) The following MATLAB code attempts to compute

$$m := \min\{\text{floating point } x : fl(1+x) > 1\},$$

under the assumption that the base $b = 2$.

```
function x = mu()

x = 1;
xp1 = x + l;
while xp1 > 1
   x = x/2 ;
   xp1 = x + 1;
end
x = 2*x;
```

On my workstation, running this code gives

```
>> mu

ans =
  2.2204e-016
```

Try this code, or a translation of it into another language, on any machines available to you. Are the answers what you expected?

(b) Under what circumstances might the code in (a) fail to compute m? (Hint: consider optimizing compilers.)

(c) What is the value of m in terms of the arithmetic parameters b and t? Note: the answer depends subtly on how rounding is performed, and in particular on whether double rounding is in effect; see Higham [550, 1991] and Moler [771, 1990]. On a workstation with an Intel 486DX chip (in which double rounding does occur), the following behaviour is observed in MATLAB:

```
>> format hex; format compact   % Hexadecimal format.

>> x = 2^(-53) + 2^ (-64) + 2^(-105) ; y = [1+x 1 x]

y =
    3ff0000000000001    3ff0000000000000    3ca0020000000001
```

```
>> x = 2^(-53) + 2^(-64) + 2^(-106); y = [1+x 1 x]

y =
   3ff0000000000000   3ff0000000000000   3ca0020000000000
```

25.2. Show that Smith's formulae (25.1) can be derived by applying Gaussian elimination with partial pivoting to the 2 x 2 linear system obtained from $(c + id)(x + iy) = a + ib$.

25.3. The following MATLAB function implements an algorithm of Malcolm [724, 1972] for determining the floating point arithmetic parameters b and t.

```
function   [beta, t] = param(x)
% a and b are floating point variables.

a = 1;
while (a+l) - a == 1
      a = 2*a;
end
b=2;
while (a+b) == a
      b = 2*b ;
end
beta =       - a;
t = 1;
a = beta;
while (a+l) - a == 1
      t = t+l;
      a= a*beta;
end
```

Run this code, or a translation into another language, on any machines available to you. Explain how the code works. (Hint: consider the integers that can be represented exactly as floating point numbers.) Under what circumstances might it fail to produce the desired results?

25.4. Hough [585, 1989] formed a 512 x 512 matrix using the following Fortran random number generator:

```
      subroutine matgen (a, lda, n, b, norms)
      REAL a(lda,1), b(1), norma
c
      init = 1325
      norms = 0.0
      do 30 j = 1,n
         do 20 i = 1,n
```

```
            init  =  mod(3125*init,65536)
            a(i,  j) = (init  -  32768  .0)/16384.0
            norms  =  max(a(i,  j),  norms)
   20     continue
   30  continue
```

He then solved a linear system using this matrix, on a workstation that uses IEEE single precision arithmetic. He found that the program took an inordinate amount of time to run and tracked the problem to underflows. On the system in question underflows are trapped and the correct result recomputed very slowly in software. Hough picks up the story:

> I started printing out the pivots in the program. They started out as normal numbers like 1 or – 10, the suddenly dropped to about 1e-7, then later to 1e-14, and then:

```
 k  82  pivot  -1.8666e-20         k  98  pivot  1.22101e-21
 k  83  pivot  -2.96595e-14        k  99  pivot  -7.12407e-22
 k  84  pivot  2.46156e-14         k  100  pivot  -1.75579e-21
 k  85  pivot  2.40541e-14         k  101  pivot  3.13343e-21
 k  86  pivot  -4.99053e-14        k  102  pivot  -6.99946e-22
 k  87  pivot  1.7579e-14          k  103  pivot  3.82048e-22
 k  88  pivot  1.69295e-14         k  104  pivot  8.05538e-22
 k  89  pivot  -1.56396e-14        k  105  pivot  -1.18164e-21
 k  90  pivot  1.37869e-14         k  106  pivot  -6.349e-22
 k  91  pivot  -3.10221e-14        k  107  pivot  -2.48245e-21
 k  92  pivot  2.35206e-14         k  108  pivot  -8.89452e-22
 k  93  pivot  1.32175e-14         k  109  pivot  -8.23235e-22
 k  94  pivot  -7.77593e-15        k  110  pivot  4.40549e-21
 k  95  pivot  1.34815e-14         k  111  pivot  1.12387e-21
 k  96  pivot  -1.02589e-21        k  112  pivot  -4.78853e-22
 k  97  pivot  4.27131e-22         k  113  pivot  4.38739e-22
                                   k  114  pivot  7.3868e-28

SIGFPE  8:  numerical  exception,  CHK,  or  TRAP
stopped  at     daxpy+0x18c:      mov1   a4@(0xe10),a3@
```

Explain this behaviour.

25.5. The version of the level-l BLAS routine `xNRM2` that is distributed with LAPACK implements the following algorithm for computing $||x||_2, x \in \mathbb{R}^n$:

```
$t = 0$
$s = 1$
for $i = 1:n$
```

```
    if $|x_i| > t$
        $s = 1 + s\,(t/x_i)^2$
        $t = |x_i|$
    else
        $s = s + (x_i/t)^2$
    end
end
$\|x\|_2 = t\sqrt{s}$
```

Prove that the algorithm works and describe its properties.

25.6. (Moler and Morrison [773, 1983], Dubrulle [323, 1983]) This MATLAB M-file computes the Pythagorean sum $\sqrt{a^2+b^2}$ using an algorithm of Moler and Morrison.

```
function p = pythag(a, b)
%PYTHAG     Pythagorean sum.

P = max(abs(a) , abs(b)) ;
q = min(abs(a) , abs(b)) ;
while 1
      r = (q/p)^2;
      if r+4 == 4, return, end
      s = r/(4+ r);
      p = p+2*s*p;
      q = s*q;
      fprintf('p = %19.15e, q = %19.15e\n', p, q)
end
```

The algorithmic immune to underflow and overflow (unless the result overflows), is very accurate, and converges in at most three iterations, assuming the unit roundoff $u \geq 10^{-20}$.

Example:

```
p = pythag(3,4); (p-5)/5

p = 4.986301369863014e+000, q = 3.698630136986301e-001
p = 4.999999974188253e+000, q = 5.080526329415358e-004
p = 5.000000000000001e+000, q = l.311372652397091e-012
ans =
  1.7764e-016
```

The purpose of this problem is to show that `pythag` is Halley's method applied to a certain equation. Halley's method for solving $f(x) = 0$ is the iteration

$$x_{n+1} = x_n - \frac{f_n/f'_n}{1 - \frac{1}{2} f_n f''_n/(f'_n)^2},$$

where f_n, f'_n and f''_n denote the values of f and its first two derivatives at x_n.

(a) For given x_0 and y_0 such that $0 < y_0 \le x_0$, the Pythagorean sum $p = \sqrt{x_0^2 + y_0^2}$ is a root of the equation $f(x) = x^2 - p^2 = 0$. Show that Halley's method applied to this equation gives

$$x_{n+1} = x_n \left(1 + 2\frac{p^2 - x_n^2}{p^2 + 3x_n^2}\right).$$

Show that if $0 \le x_0 \le p = \sqrt{x_0^2 + y_0^2}$ then $x_n \le x_{n+1} \le p$ for all n. Deduce that $y_n := \sqrt{p^2 - x_n^2}$ is defined and that

$$x_{n+1} = x_n \left(1 + 2\frac{y_n^2}{4x_n^2 + y_n^2}\right),$$

$$y_{n+1} = \frac{y_n^3}{4x_n^2 + y_n^2}.$$

Confirm that pythag implements a scaled version of these equations.

(b) Show that

$$p - x_{n+1} = \frac{(p - x_n)^3}{p^2 + 3x_n^2},$$

which displays the cubic convergence of the iteration.

25.7. (Incertis [604, 1985]) (a) Given a skew-symmetric matrix $Y \in \mathbb{R}^{n\times n}$ with $\|Y\|_2 \le 1$, show that there is a real, symmetric X satisfying $X^2 = I + Y^2$ such that X + Y is orthogonal.

(b) Consider the following iteration, for P_0, $Q_0 \in \mathbb{R}^{n\times n}$ with $\|P_0\|_2 \ge \|Q_0\|_2$, which generalizes to matrices the "pythag" iteration in Problem 25.6.

```
for $k = 1:\infty$
    $R_k = (Q_k P_k^{-1})^2$
    $S_k = R_k(4I + R_k)^{-1}$
    $P_{k+1} = P_k + 2S_k P_k$
    $Q_{k+1} = S_k Q_k$
end
```

Show that this iteration can be used to compute X in part (a)

(c) Investigate the convergence of the iteration in (b) for general P_0 and Q_0.

25.8. Why might we prefer the expression $\sqrt{|x|}/\sqrt{3}$ to $\sqrt{|x|/3}$ in an algorithm intended to be robust in floating point arithmetic?

Chapter 26
A Gallery of Test Matrices

Many tricks or treats associated with the Hilbert matrix
may seem rather frightening or fascinating.

— MAN-DUEN CHOI, *Tricks or Treats with the Hilbert Matrix* (1983)

I start by looking at a 2 by 2 matrix.
Sometimes I look at a 4 by 4 matrix.
That's when things get out of control and too hard.
Usually 2 by 2 or 3 by 3 is enough, and I look at them,
and I compute with them, and I try to guess the facts.
First, think of a question.
Second, I look at examples, and then third,
guess the facts.

— PAUL R. HALMOS[24] (1991)

When people look down on matrices,
remind them of great mathematicians such as
Frobenius, Schur, C. L. Siegel, Ostrowski, Motzkin, Kac, etc.,
who made important contributions to the subject.

— OLGA TAUSSKY, *How I Became a Torchbearer for Matrix Theory (1988)*

---

[24] From interviews by Albers in [9, 1991].

Ever since the first computer programs for matrix computations were written in the 1940s, researchers have been devising matrices suitable for test purposes and investigating the properties of these matrices. In the 1950s and 1960s it was common for a whole paper to be devoted to a particular test matrix: typically its inverse or eigenvalues would be obtained in closed form.

Early collections of test matrices include those of Newman and Todd [796, 1958] and Rutishauser [886, 1968]; most of Rutishauser's matrices come from continued fractions or moment problems. Two well-known books present collections of test matrices. Gregory and Karney [482, 1969] deal exclusively with the topic, while Westlake [1076, 1968] gives an appendix of test matrices.

In this chapter we present a gallery of matrices. We describe their properties and explain why they are useful (or not so useful, as the case may be) for test purposes. The coverage is limited. A comprehensive, up-to-date, and well-documented collection of parametrized test matrices may be found in the Test Matrix Toolbox, described in Appendix E. Of course, MATLAB itself contains a number of special matrices that can be used for test purposes (type `help specmat`).

Several other types of matrices would have been included in this chapter had they not been discussed elsewhere in the book. These include magic squares (Problem 6.4), the Kahan matrix (8.10), Hadamard matrices (§9.3), and Vandermonde matrices (Chapter 21).

The matrices described here can be modified in various ways while still preserving some or all of their interesting properties. Among the many ways of constructing new test matrices from old are

- Similarity transformations $A \leftarrow X^{-1}AX$.
- Unitary transformations $A \leftarrow UAV$, where $U^*U = V^*V = I$.
- Kronecker products $A \leftarrow A \otimes B$ or $B \otimes A$ (for which MATLAB has a routine `kron`).
- Powers $A \leftarrow A^k$.

26.1. The Hilbert and Cauchy Matrices

The Hilbert matrix $H_n \in \mathbb{R}^{n\times n}$ with elements $h_{ij} = 1/(i+j-1)$, is perhaps the most famous of all test matrices. It was widely used in the 1950s and 1960s for testing inversion algorithms and linear equation solvers. Its attractions were threefold: it is very ill conditioned for even moderate values of n, formulae are known for the elements of the inverse, and the matrix arises in a practical problem: least squares fitting by a polynomial expressed in the monomial basis.

Despite its past popularity and notoriety, the Hilbert matrix is not a good test matrix. It is too special. Not only is it symmetric positive definite, but it is totally positive. This means, for example, that Gaussian elimination without pivoting is guaranteed to produce a small componentwise relative backward error (as is Cholesky factorization). Thus the Hilbert matrix is not a typical ill-conditioned matrix.

The (i, j) element of the inverse of the Hilbert matrix H_n is the integer

$$(-1)^{i+j}(i+j-1)\binom{n+i-1}{n-j}\binom{n+j-1}{n-i}\binom{i+j-2}{i-1}^2, \qquad (26.1)$$

and

$$\det(H_n) = \frac{(1!\,2!\ldots(n-1)!\,)^4}{(1!\,2!\ldots(2n-1)!\,)} \sim 2^{-2n^2}. \qquad (26.2)$$

There are many ways to rewrite the formula (26.1). These formulae are best obtained as special cases of those for the Cauchy matrix below.

The Cholesky factor R_n of the inverse of the Hilbert matrix is known explicitly, as is R_n^{-1}:

$$r_{ij} = \frac{\sqrt{2i-1}\,[(j-1)!]^2}{(i+j-1)!(j-i)!} \qquad j \ge i, \qquad (26.3)$$

$$(R_n^{-1})_{ij} = (-1)^{i+j}\sqrt{2j-1}\binom{i+j-2}{i-1}\binom{j-1}{i-1} \qquad j \ge i, \qquad (26.4)$$

One interesting application of these formulae is to compute the eigenvalues of H_n as the squares of the singular values of R_n; if R_n is evaluated from (26.3) and the one-sided Jacobi algorithm is used to compute the singular values then high relative accuracy is obtained for every eigenvalue, as shown by Mathias [732, 1995].

The condition number of the Hilbert matrix grows at an exponential rate: $k_2(H_n) \sim \exp(3.5n)$ [1004, 1954]. See Table 26.1 for the first few condition numbers (these were obtained by computing the inverse exactly using MATLAB'S Symbolic Math Toolbox [204, 1993] and then computing the norm of the numeric representation of the inverse; the numbers given are correct to the figures shown).

It is an interesting fact that the matrix $\widetilde{H}_n = (1/(i+j)) \in \mathbb{R}^{n\times n}$ (a submatrix of H_{n+1}) satisfies $m_n := \|\widetilde{H}_n\|_2 = \pi + O(1/\log n)$ as $n \to \infty$, as proved by Taussky [996, 1949]. The convergence to π is very slow: $m_{200} = 2.01$, $m_{300} = 2.08$, $m_{400} = 2.12$.

That H_n^{-1} is known explicitly is not as useful a property for testing an inversion algorithm as it might appear, because H_n cannot be stored exactly in floating point arithmetic. This means that when we attempt to invert H_n we at best invert $H_n + \Delta H$ (the matrix actually stored on the computer),

Table 26.1. *Condition numbers of Hilbert and Pascal matrices.*

n	$\kappa_2(H_n)$	$\kappa_2(P_n)$
2	1.9281e1	6.8541e0
3	5.2406e2	6.1984e1
4	1.5514e4	6.9194e2
5	4.7661e5	8.5175e3
6	1.4951e7	1.1079e5
7	4.7537e8	1.4934e6
8	1.5258e10	2.0645e7
9	4.9315e11	2.9078e8
10	1.6026e13	4.1552e9
11	5.2307e14	6.0064e10
12	1.7132e16	8.7639e11
13	5.6279e17	1.2888e13
14	1.8534e19	1.9076e14
15	6.1166e20	2.8396e15
16	2.0223e22	4.2476e16

where $|\Delta H| \le uH_n$, and $(H_n + \Delta H)^{-1}$ can differ greatly from H_n^{-1}, in view of the ill conditioning. A possible way round this difficulty is to start with the integer matrix H_n^{-1}, but its entries are so large that they are exactly representable in IEEE double precision arithmetic only for n less than 13.

The Hilbert matrix is a special case of a Cauchy matrix $C_n \in \mathbb{R}^{n\times n}$, whose elements are $c_{ij} = 1/(x_i + y_j)$, where $x, y \in \mathbb{R}^n$ are given n-vectors (take $x_i = y_i = i - 0.5$ for the Hilbert matrix). The following formulae give the inverse and determinant of C_n, and therefore generalize those for the Hilbert matrix. The (i, j) element of C_n^{-1} is

$$\frac{\displaystyle\prod_{1\le k\le n} (x_j + y_k)(x_k + y_i)}{(x_j + y_i) \displaystyle\prod_{\substack{1\le k\le n\\ k\ne j}} (x_j - x_k) \prod_{\substack{1\le k\le n\\ k\ne i}} (y_i - y_k)}$$

and

$$\det(C_n) = \frac{\displaystyle\prod_{1\le i<j\le n} (x_j - x_i)(y_j - y_i)}{\displaystyle\prod_{1\le i,j\le n} (x_i + y_j)},$$

the latter formula having been published by Cauchy in 1841 [189, 1841, pp. 151–159]. The LDU factors of C_n have been found explicitly by Gohberg and Koltracht [454, 1990]: $l_{kk} = u_{kk} \equiv 1$ and

$$l_{jk} = \prod_{m=1}^{k} \frac{x_k + y_m}{x_j + y_m} \prod_{m=1}^{k-1} \frac{y_j - y_m}{y_k - y_m}, \qquad 1 \le k < j \le n,$$

$$d_{kk}^{-1} = (-1)^{k-1}(x_k + y_k) \prod_{m=1}^{k-1} \frac{(y_k + x_m)(y_m + x_k)}{(y_k - y_m)(x_m - x_k)}, \qquad k = 1\!:\!n,$$

$$u_{kj} = \prod_{m=1}^{k} \frac{y_k + x_m}{y_j + x_m} \prod_{m=1}^{k-1} \frac{y_j - y_m}{y_k - y_m}, \qquad 1 \le k < j \le n.$$

It is known that C_n is totally positive if $0 < x_1 < \cdots < x_n$ and $0 < y_1 < \cdots < y_n$ (as is true for the Hilbert matrix) [998, 1962, p. 295]. Interestingly, the sum of all the elements of C_n^{-1} is $\sum_{i=1}^{n}(x_i + y_i)$ [667, 1973, Ex. 44, §1.2.3].

26.2. Random Matrices

Random matrices are widely used for test purposes. Perhaps the earliest use of random matrices in numerical analysis was by von Neumann and Goldstine [1057, 1947], [462, 1951], who estimated the size of their error bounds for matrix inversion (see §9.6) for the case of random matrices; to do this, they had to estimate the condition number of a random matrix.

Intuitively, one might expect random matrices to be good at revealing programming errors and unusual behaviour of algorithms, but this expectation is not necessarily correct. For example, Miller [759, 1984, pp. 96-97] describes a mutation experiment involving Fortran codes for Gaussian elimination without pivoting, Gaussian elimination with partial pivoting, and Gauss–Jordan elimination with partial pivoting. For each code, all possible mutants were generated, where a mutant is obtained by making a single typographical change to the source code. All the mutants were tested on a single random linear system $Ax = b$, with known solution, where a_{ij} was chosen from the uniform [– 1, 1] distribution. Many mutants were detected by their failure to pass the test of producing a solution with forward error less than a tolerance. However, some mutants passed this test, including all those that solve a system correctly in exact arithmetic; mutants in the latter class include those that select an incorrect pivot row and thus implement a numerically unstable algorithm. A conclusion to be drawn from Miller's experiment is that random test data can reveal some programming errors, but will not reveal all.

A good example of a problem for which random test matrices are very poor at revealing algorithmic weaknesses is matrix condition number estimation. The popular condition estimation algorithms can yield poor estimates but, in practice, never produce them for a random matrix (see Chapter 14). The role of random matrices here is to indicate the *average* quality of the estimates.

Edelman [340, 1993] summarizes the properties of random matrices well when he says that

> What is a mistake is to psychologically link a random matrix with the intuitive notion of a "typical" matrix or the vague concept of "any old matrix." In contrast, we argue that "random matrices" are very *special* matrices. The larger the size of the matrices the more predictable they are because of the central limit theorem.

Various results are known about the behaviour of matrices with elements from the normal N(0, 1) distribution. Matrices of this type are generated by MATLAB'S `randn` function. Let A_n denote an $n \times n$ matrix from this distribution and let E(·) be the expectation operator. Then, in the appropriate probabilistic sense, the following results hold as $n \rightarrow \infty$:

$$E(\log(\kappa_2(A_n))) \approx \log n + 1.537 \qquad \text{(real data)}, \tag{26.5}$$

$$E(\log(\kappa_2(A_n))) \approx \log n + 0.982 \qquad \text{(complex data)}, \tag{26.6}$$

$$\|A_n\|_2 \approx 2\sqrt{n} \qquad \text{(real data)}, \tag{26.7}$$

$$\|A_n\|_2 \approx 2\sqrt{2}\sqrt{n} \qquad \text{(complex data)}, \tag{26.8}$$

$$\rho(A_n) \approx \sqrt{n}. \qquad \text{(real or complex data)}. \tag{26.9}$$

For details of (26.5)-(26.8) see Edelman [335, 1988]. Edelman conjectures that the condition number results are true for any distribution with mean 0—in particular, the uniform [– 1, 1] distribution used by MATLAB'S `rand` function. The results (26.5) and (26.6) show that random *matrices from the normal* $N(0, 1)$ *distribution tend to be very well conditioned.*

The spectral radius result (26.9) has been proved as an inequality by Geman [432, 1986] for independent identically distributed random variables a_{ij} with zero mean and unit variance, and computer experiments suggest the approximate equality for the standard normal distribution [432, 1986].

A question of interest in eigenvalue applications is how many eigenvalues of a random real matrix are real. The answer has been given by Edelman, Kostlan, and Shub [344, 1994]: denoting by E_n the expected number of real eigenvalues of an $n \times n$ matrix from the normal $N(0, 1)$ distribution,

$$\lim_{n\rightarrow\infty} \frac{E_n}{\sqrt{n}} = \sqrt{\frac{2}{\pi}}.$$

Thus the proportion of real eigenvalues, E_n/n, tends to zero as $n \rightarrow \infty$. Exact formulae for E_n for finite n are also given in [344, 1994].

26.3. "Randsvd" Matrices

By randsvd[25] we mean a matrix $A \in \mathbb{R}^{m\times n}$ formed as $A = USV^T$, where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are random orthogonal matrices and $S = \mathrm{diag}(s_i)$ $(\sigma_i \geq 0)$ is a given matrix of singular values. This type of matrix has a predetermined singular value distribution (and 2-norm condition number), but is, nevertheless, random.

`Randsvd` matrices have been widely used as test matrices, for example for condition estimators [534, 1987], [946, 1980], and in the LAPACK testing software. Singular value distributions of interest include, with $\alpha := {}_{K2}(A) \geq 1$ **a** parameter,

1. one large singular value: $s_1 = 1$, $s_i = \alpha^{-1}$, $i = 2{:}n$;
2. one small singular value: $s_i = 1$, $i = 1{:}n - 1$, $s_n = \alpha^{-1}$;
3. geometrically distributed singular values: $s_i = b^{1-i}$, $i = 1{:}n$, where $\beta = \alpha^{1/(n-1)}$;
4. arithmetically distributed singular values: $s_i = 1 - (1 - \alpha^{-1})(i - 1)/(n - 1)$, $i = 1{:}n$.

To be precise about what we mean by "random orthogonal matrix" we specify matrices from the Haar distribution, which is a natural distribution over the space of orthogonal matrices, defined in terms of a measure called the Haar measure [780, 1982, 52.1.4]. If $A \in \mathbb{R}^{n\times n}$ has elements from the normal $N(0, s^2)$ distribution and A has the QR factorization $A = QR$, with the factorization normalized so that the diagonal elements of R are positive, then Q is from the Haar distribution, for any variance s^2 [100, 1979], [946, 1980]. If we compute Q from this prescription, the cost is $2n^3$ flops. For our `randsvd` application, a more efficient approach is based on the following result of Stewart [946, 1980].

Theorem 26.1 (Stewart). *Let the independent vectors $x_i \in \mathbb{R}^i$ have elements from the normal* $N(0, 1)$ *distribution for* $i = 1{:}n - 1$. *Let* $P_i = \mathrm{diag}(I_{i-1}, \overline{P}_i)$, *where* $\overline{P}_i$ *is the Householder transformation that reduces* x_i *to* $r_{ii}e_l$. *Then the product* $Q = DP_1P_2 \ldots P_{n-l}$ *is a random orthogonal matrix from the Haar distribution, where* $D = \mathrm{diag}(\mathrm{sign}(r_{ii}))$. □

This result allows us to compute a product form representation of a random $n \times n$ orthogonal matrix from the Haar distribution in $O(n^2)$ flops. If we implicitly compute $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ using the construction in Theorem 26.1, and then form $A = USV^T$, exploiting the structure, the cost is $m^3 + n^3$ flops. The alternative, which involves obtaining U and V from the QR factorization of random matrices A, is about twice as expensive.

---

[25] Randsvd is the name of the MATLAB M-file in the Test Matrix Toolbox that generates matrices of this type.

Note that forming USV^T with U and V *single* Householder matrices, as is sometimes done in the literature, is not recommended, as it produces matrices of a very special form: diagonal plus a rank-2 correction.

The construction of `randsvd` matrices is, of course, easily adapted to produce random symmetric matrices $A = QLQ^T$ with given eigenvalues.

26.4. The Pascal Matrix

> *The numbers in Pascal's triangle satisfy, practically speaking, infinitely many identities.*
>
> — RONALD L. GRAHAM, DONALD E. KNUTH, and OREN PATASHNIK, *Concrete Mathematics* (1989)

A particularly entertaining test matrix is the Pascal matrix $P_n \in \mathbb{R}^{n\times n}$, defined by

$$p_{ij} = \frac{(i+j-2)!}{(i-1)!(j-1)!} = \binom{i+j-2}{j-1}.$$

The rows of Pascal's triangle appear as anti-diagonals of P_n[26]:

```
>> p = pascal(6)

P =
     1     1     1     1     1     1
     1     2     3     4     5     6
     1     3     6    10    15    21
     1     4    10    20    35    56
     1     5    15    35    70   126
     1     6    21    56   126   252
```

The earliest references to the Pascal matrix appear to be in 1958 by Newman and Todd [796, 1958] and by Rutishauser [887, 1958] (see also Newman [795, 1962, pp. 240-24 1]); Newman and Todd say that the matrix was introduced to them by Rutishauser. The matrix was independently suggested as a test matrix by Caffney [177, 1963].

Rutishauser [886, 1968, §8] notes that P_n belongs to the class of moment matrices M whose elements are contour integrals

$$m_{ij} = \int_C z^{j-1}(\overline{z})^{i-1} w(z)\, dz.$$

---

[26] In the MATLAB displays below we use the pascal function from the Test Matrix Toolbox. This differs from the pascal function supplied with Matlab 4.2 only in that pascal (n, 2) is rearranged.

All moment matrices corresponding to a positive weight function $w(z)$ on a contour C are Hermitian positive definite (as is easily verified by considering the quadratic form y^*My). The choice $C = [0, 1]$, with weight function $w(z) = 1$, yields the Hilbert matrix. The Pascal matrix is obtained for C the circle $\{z : |z-1| = 1\}$ and $w(z) = (2\pi i(z-1))^{-1}$ (not $w(z) = (2\pi)^{-1}$ as stated in [886, 1968]); the change of variable $z = 1 + \exp(iq)$ yields a moment integral with a positive weight function.

Remarkably, the Cholesky factor of the Pascal matrix again contains the rows of Pascal's triangle, now arranged columnwise:

```
>> R = chol(P)

R =
     1     1     1     1     1     1
     0     1     2     3     4     5
     0     0     1     3     6    10
     0     0     0     1     4    10
     0     0     0     0     1     5
     0     0     0     0     0     1
```

The scaled and transposed Cholesky factor $S = R^T \operatorname{diag}(1, -1, 1, -1, \ldots, (-1)^{n+1})$ is returned by `pascal (n, 1 )`:

```
>> S = pascal(6, 1)

S =
     1     0     0     0     0     0
     1    -1     0     0     0     0
     1    -2     1     0     0     0
     1    -3     3    -1     0     0
     1    -4     6    -4     1     0
     1    -5    10   -10     5    -1
```

It is involuntary: $S^2 = I$. This special property leads us to several more properties of $P = P_n$. First, since $P = SS^T$, $P^{-1} = S^{-T}S^{-1} = S^TS$, and so P^{-1} has integer entries (as is also clear from the fact that $\det(P) = \det(R)^2 = 1$). Moreover,

$$P = SS^T = S(S^TS)S^{-1} = SP^{-1}S^{-1},$$

so P and P^{-1} are similar and hence have the same eigenvalues. In other words, the eigenvalues appear in reciprocal pairs. In fact, the characteristic polynomial π_n has a palindromic coefficient vector, which implies the reciprocal eigenvalues property, since $\pi_n(\lambda) = \lambda^n \pi_n(1/\lambda)$. This is illustrated as follows (making use of MATLAB'S Symbolic Math Toolbox):

```
>> charpoly(P)

ans =
1-351*x+6084*x^2-13869*x^3+6084*x^4-351*x^5+x^6

>> eig(P)

ans =
   0.0030
   0.0643
   0.4893
   2.0436
  15.5535
 332.8463
```

Since P is symmetric, its eigenvalues are its singular values and so we also have that $\|P\|_2 = \|P^{-1}\|_2$ and $\|P\|_F = \|P^{-1}\|_F$. Now

$$p_{nn} \le \|P\|_2 \le (\|P\|_1\|P\|_\infty)^{1/2} = \binom{2n-1}{n} p_{nn},$$

where for the last equality we used a binomial coefficient summation identity from [477, 1989, p. 161]. Hence, using Stirling's approximation ($n! \sim \sqrt{2\pi n}(n/e)^n$),

$$\kappa_2(P_n) \sim \left(\frac{(2n)!}{(n!)^2}\right)^2 \sim \frac{16^n}{n\pi}.$$

Thus P_n is exponentially ill conditioned as $n \to \infty$.

It is worth pointing out that it is not hard to generate symmetric positive definite matrices with determinant 1 and the reciprocal root property. Let $X = ZDZ^{-1}$ where Z is nonsingular and $D = \mathrm{diag}(\pm 1) \ne \pm I$. Then $X^2 = I$ and the matrix $A = X^TX$ has the desired properties. If we choose Z lower triangular then X is the Cholesky factor of A up to a column scaling by diag(±1).

The inverse of the Pascal matrix was found explicitly by Cohen [230, 1975]: the (i,j) element is

$$(-1)^{i-j}\sum_{k=0}^{n-i}\binom{i+k-1}{k}\binom{i+k-1}{i+k-j}, \qquad i \ge j.$$

The Pascal matrix can be made singular simply by subtracting 1 from the (n,n) element. To see this, note that

$$\begin{aligned} P - e_ne_n^T &= SS^T - e_ne_n^T = S(I-(Se_n)(Se_n)^T)S^T \\ &= S\,\mathrm{diag}(1,1,\ldots,1,0)S^T. \end{aligned}$$

This perturbation, $\Delta P = -e_n e_n^T$, is far from being the smallest one that makes P singular, which is $\Delta P_{opt} = -\lambda_n v_n v_n^T$, where λ_n is the smallest eigenvalue of P and v_n is the corresponding unit eigenvector, for $\|\Delta P_{\mathrm{opt}}\|_2 = \lambda_n = \|P^{-1}\|_2^{-1}$ is of order $(n!)^2/(2n)! \sim 4^{-n}\sqrt{n\pi}$, as we saw above.

A more subtle property of the Pascal matrix is that it is totally positive. Karlin [644, 1968, p. 137] shows that the matrix with elements $\binom{i+j-1}{j}$, $(i, j = 0,1, \ldots)$ is totally positive; the Pascal matrix is a submatrix of this one and hence is also totally positive. From the total positivity it follows that the Pascal matrix has distinct eigenvalues, which (as we already know from the positive definiteness) are real and positive, and that its ith eigenvector (corresponding to the ith largest eigenvalue) has exactly $i - 1$ sign changes [414, 1959, Thin. 13, p. 105].

T = `pascal (n, 2)` is obtained by rotating S clockwise through 90 degrees and multiplying by — 1 if n is even:

```
>> T = pascal(6, 2)

T =
    -1    -1    -1    -1    -1    -1
     5     4     3     2     1     0
   -10    -6    -3    -1     0     0
    10     4     1     0     0     0
    -5    -1     0     0     0     0
     1     0     0     0     0     0
```

It has the surprising property that it is a cube root of the identity, a property noted by Turnbull [1028, 1929, p. 332]:

```
>> T*T

ans =
     0     0     0     0     0     1
     0     0     0     0    -1    -5
     0     0     0     1     4    10
     0     0    -1    -3    -6   -lo
     0     1     2     3     4     5
    -1    -1    -1    -1    -1    -1

>> T*T*T

ans =
     1     0     0     0     0     0
     0     1     0     0     0     0
     0     0     1     0     0     0
```

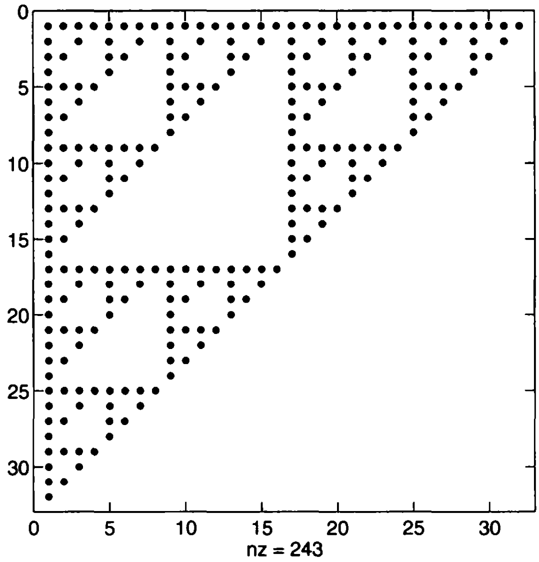


Figure 26.1. spy(rem(pascal(32),2)).

```
0    0    0    1    0    0
0    0    0    0    1    0
0    0    0    0    0    1
```

Finally, we note that it is trivial to plot an approximation to the Sierpinski gasket [824, 1992, §2.2], [478,1992] in MATLAB: simply type spy(rem(pascal (n), 2)). See Figure 26.1. The picture produced by this command is incorrect for large n, however, because the elements of pascal(n) become too large to be exactly representable in floating point arithmetic.

26.5. Tridiagonal Toeplitz Matrices

A tridiagonal Toeplitz matrix has the form

$$T_n(c,d,e) = \begin{bmatrix} d & e & & \\ c & d & \ddots & \\ & \ddots & \ddots & e \\ & & c & d \end{bmatrix}.$$

Such matrices arise, for example, when discretizing partial differential equations or boundary value problems for ordinary differential equations. The eigenvalues are known explicitly [884, 1947], [885, 1952], [1006, 1977, pp. 155–156]:

$$d + 2(ce)^{1/2}\cos(k\pi/(n+1)), \qquad k = 1{:}n.$$

The eigenvalues are also known for certain variations of the symmetric matrix $T_n(c, d, c)$ in which the (1,1) and (n, n) elements are modified; see Gregory and Karney [482, 1969].

The matrix $T_n(-1, 2, -1)$ is minus the well-known second difference matrix, which arises in applying central differences to a second derivative operator. Its inverse has (i, j) element $-i(n - j + 1)/(n + 1)$ (cf. Theorem 14.9). The condition number satisfies $\kappa_2(T_n) \sim (4/\pi^2)n^2$.

One interesting property of $T_n(c, d, e)$ is that the diagonals of its LU factorization converge as $n \to \infty$ when T_n is symmetric and diagonally dominant, and this allows some savings in the computation of the LU factorization, as shown by Malcolm and Palmer [725, 1974]. Similar properties hold for cyclic reduction; see Bondeli and Gander [134, 1994].

26.6. Companion Matrices

The companion matrix associated with the characteristic polynomial

$$\det(A - \lambda I) = (-1)^n(\lambda^n - a_{n-1}\lambda^{n-1} - \cdots - a_0)$$

of $A \in \mathbb{C}^{n\times n}$ is the matrix

$$C = \begin{bmatrix} a_{n-1} & a_{n-2} & \dots & \dots & a_0 \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & & 0 \\ \vdots & & \ddots & 0 & 0 \\ 0 & \dots & \dots & 1 & 0 \end{bmatrix}.$$

The Test Matrix Toolbox function `compan` computes C, via the call `C = compan (A)`, It is easy to check that C has the same characteristic polynomial as A, and that if λ is an eigenvalue of C then $[\lambda^{n-1}, \lambda^{n-2}, \dots, \lambda, 1]^T$ is a corresponding eigenvector. Since $C - \lambda I$ has rank at least $n - 1$ for any λ, C is nonderogatory, that is, in the Jordan form no eigenvalue appears in more than one Jordan block. It follows that A is similar to C only if A is nonderogatory.

There are no explicit formulae for the eigenvalues of C, but, perhaps surprisingly, the singular values have simple representations, as found by Kenney and Laub [651, 1988] (see also Kittaneh [660, 1995]):

$$\sigma_1^2 = \frac{1}{2}\left(\alpha + \sqrt{\alpha^2 - 4a_0^2}\right),$$

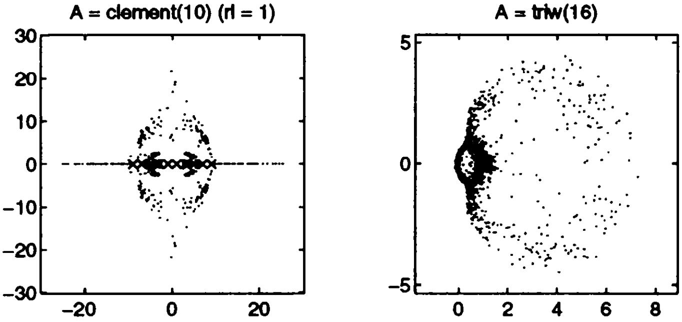


Figure 26.2. *Pseudospectra of* compan(A).

$$\sigma_i^2 = 1, \qquad i = 2\colon n-1,$$
$$\sigma_n^2 = \frac{1}{2}\left(\alpha - \sqrt{\alpha^2 - 4a_0^2}\right),$$

where $\alpha = 1 + a_0^2 + \cdots + a_{n-1}^2$.

The compan function is a useful means for generating new test matrices from old. For any $A \in \mathbb{C}^{n\times n}$, compan(A) is a nonnormal matrix with the same eigenvalues as A (to be precise, compan(A) is normal if and only if $a_0 = 1$ and $a_i = 0$ for $i > 0$).

Companion matrices tend to have interesting pseudospectra, as illustrated in Figure 26.2. For more information on the pseudospectra of companion matrices see Toh and Trefethen [1007, 1994].

26.7. Notes and References

The earliest reference to the Hilbert matrix appears to be [570, 1894], wherein Hilbert obtains the formula (26.2) for $\det(H_n)$.

The formula (26.1) for H_n^{-1} is taken from Choi [207, 1983], who describes various interesting properties of the Hilbert matrix and its infinite analogue. An excellent reference for the derivation of formulae for the inverse of the Cauchy and Hilbert matrices is Knuth [667, 1973, pp. 35–37]. Another reference for the Cauchy matrix is Tyrtyshnikov [1034, 1991]. The formulae (26.3) and (26.4) for the Cholesky factor and its inverse are from Choi [207, 1983] and Todd [1004, 1954].

Forsythe and Moler [396, 1967] have a chapter devoted to the Hilbert matrix, in which they describe the underlying least squares problem and discuss numerical computation of the inverse. There have been many papers on the

Hilbert matrix; two of particular interest are by Todd [1004, 1954], [1005, 1961].

Other references on the eigenvalues and condition numbers of random matrices include Edelman [336, 1991], [339, 1992] and Kostlan [671, 1992].

Anderson, Olkin, and Underhill [20, 1987] suggest another way to construct random orthogonal matrices from the Haar distribution, based on products of random Givens rotations. Marsaglia and Olkin [728, 1984] discuss the generation of random correlation matrices (symmetric positive semidefinite matrices with ones on the diagonal).

The involuntary triangular matrix `pascal (n, 1)` arises in the step-size changing mechanism in an ordinary differential equation code based on backward differentiation formulae; see Shampine and Reichelt [912, 1995].

We mention some other collections of test matrices. The Harwell-Boeing collection of sparse matrices, largely drawn from practical problems, is presented by Duff, Grimes, and Lewis [327, 1989], [328, 1992]. Bai [36, 1994] is building a collection of test matrices for large-scale nonsymmetric eigenvalue problems. Zielke [1128, 1986] gives various parametrized rectangular matrices of fixed dimension with known generalized inverses. Demmel and McKenney [299, 1989] present a suite of Fortran 77 codes for generating random square and rectangular matrices with prescribed singular values, eigenvalues, band structure, and other properties. This suite was the inspiration for the `randsvd` routine in the Test Matrix Toolbox and is part of the testing code for LAPACK (see below).

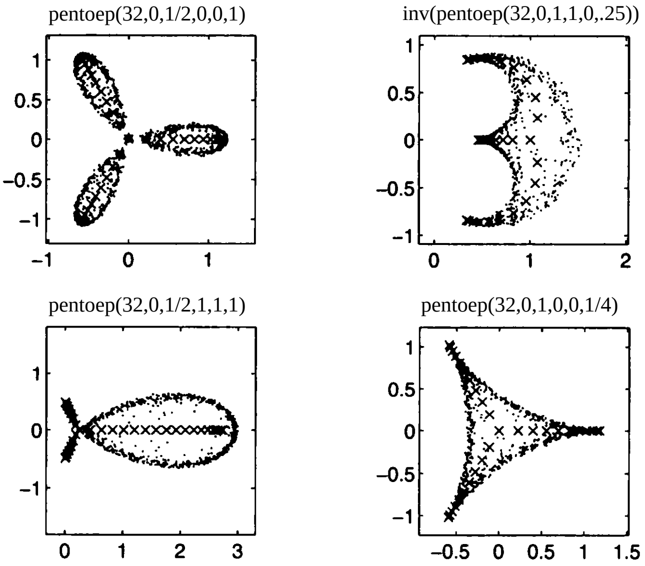
26.7.1. LAPACK

The LAPACK distribution contains a suite of routines for generating test matrices, located in the directory `LAPACK/TESTING/MATGEN` (in Unix notation). These routines (whose names are of the form `xLAxxx`) are used for testing when LAPACK is installed and are not described in the *LAPACK Users' Guide* [17, 1995].

Problems

26.1. Investigate the spectral and pseudospectral properties of pentadiagonal Toeplitz matrices. See Figure 26.3 (`pentoep` is a function in the Test Matrix Toolbox that generates matrices of this type). References are Beam and Warming [85, 1993] and Reichel and Trefethen [866, 1992].

26.2. (RESEARCH PROBLEM) Two methods for generating a real orthogonal matrix from the Haar distribution are Stewart's method (Theorem 26.1), based on Householder transformations, and a method of Anderson, Olkin, and Underhill [20, 1987], based on Givens rotations. Compare the efficiency


pentoep(32,0,1/2,0,0,1)

inv(pentoep(32,0,1,1,0,.25))

pentoep(32,0,1/2,1,1,1)

pentoep(32,0,1,0,0,1/4)

Figure 26.3. *Pseudospectm of 32 x 32 pentadiagonal Toeplitz matrices.*

of these two methods when used to generate randsvd matrices. Investigate the use of the Givens rotations approach to construct random banded matrices with given singular values and random symmetric banded matrices with given eigenvalues, and compare with the technique of generating a full matrix and then using band reduction (as implemented in the routine randsvd in the Test Matrix Toolbox).

26.3. (RESEARCH PROBLEM) Develop an efficient algorithm for computing a unit upper triangular n x *n* matrix with prescribed singular values $s_1, ... s_n$, where $\prod_i \sigma_i = 1$.

Appendix A
Solutions to Problems

1.1 Since $\widetilde{E}_{\mathrm{rel}}(\widehat{x}) = E_{\mathrm{rel}}(\widehat{x})|x|/|\widehat{x}|$ and $\widehat{x} = x(1 \pm E_{\mathrm{rel}}(\widehat{x}))$, we have

$$\frac{E_{\mathrm{rel}}(\widehat{x})}{1 + E_{\mathrm{rel}}(\widehat{x})} \le \widetilde{E}_{\mathrm{rel}}(\widehat{x}) \le \frac{E_{\mathrm{rel}}(\widehat{x})}{1 - E_{\mathrm{rel}}(\widehat{x})}.$$

Hence if $E_{\mathrm{rel}}(\widehat{x}) < 0.01$, say, then there is no difference between E_{rel} and $\widetilde{E}_{\mathrm{rel}}$ for practical purposes.

1.2. Nothing can be concluded about the last digit before the decimal point. Evaluating y to higher precision yields

t	y
35	262537412640768743.99999999999925007
40	262537412640768743.9999999999992500725972

This shows that the required digit is, in fact, 3. The interesting fact that y is so close to an integer was pointed out by Lehmer [697, 1943], who explains its connection with number theory. It is known that y is irrational [959, 1991].

1.3.

1. $x/(\sqrt{x+1} + 1)$.
2. $2\sin\frac{x-y}{2}\cos\frac{x+y}{2}$.
3. $(x - y)(x + y)$. Cancellation has not been avoided, but it is now harmless if x and y are known exactly (see also Problem 3.9).
4. $\sin x/(1 + \cos x)$.
5. $c = ((a - b)^2 + ab(2\sin q/2)^2)^{1/2}$.

1.4. $a + ib = (x + iy)^2 = x^2 - y^2 + 2ixy$, so $b = 2xy$ and $a = x^2 - y^2$, giving $x^2 - b^2/(4x^2) = a$, or $4x^4 - 4ax^2 - b^2 = 0$. Hence

$$x^2 = \frac{4a \pm \sqrt{16a^2 + 16b^2}}{8} = \frac{a \pm \sqrt{a^2 + b^2}}{2}.$$

If $a \ge 0$ we use this formula with the plus sign, since $x^2 \ge 0$. If $a < 0$ this formula is potentially unstable, so we use the rewritten form

$$x^2 = \frac{b^2}{2(-a + \sqrt{a^2 + b^2})}.$$

In either case we get two values for x and recover y from $y = b/(2x)$. Note that there are other issues to consider here, such as scaling to avoid overflow.

1.5. We need a way to compute $f(x) = \log(1 + x)$ accurately for all $x \geq 0$. A straightforward evaluation of $\log(1 + x)$ is not sufficient, since the addition $1 + x$ loses significant digits of x when x is small. The following method is effective (for another approach, see Hull, Fairgrieve, and Tang [592, 1994]): calculate $w = 1 + x$ and then compute

$$f(x) = \begin{cases} x, & w = 1, \\ \dfrac{x \log w}{w - 1}, & w \neq 1. \end{cases}$$

The explanation of why this method works is similar to that for the method in §1.14.1. We have $\widehat{w} = (1 + x)(1 + \delta)$, $|\delta| \leq u$, and if $\widehat{w} = 1$ then $|x| \leq u + u^2 + u^3 + \cdots$, so from the Taylor series $f(x) = x(1 - x/2 + x^2/3 - \cdots)$ we see that $f(x) = x$ is a correctly rounded result. If $\widehat{w} \neq 1$ then

$$\widehat{f} = \frac{(x \log \widehat{w})(1 + \epsilon_1)(1 + \epsilon_2)}{(\widehat{w} - 1)(1 + \epsilon_3)}(1 + \epsilon_4), \qquad |\epsilon_i| \leq u.$$

Defining $\widehat{w} =: 1 + z$,

$$g(\widehat{w}) := \frac{\log \widehat{w}}{\widehat{w} - 1} = \frac{\log(1 + z)}{z} = 1 - z/2 + z^2/3 - \cdots,$$

so if $\approx w \approx 1$ (thus $x \approx 0$) then

$$g(w) - g(\widehat{w}) \approx \frac{\widehat{w} - w}{2} = \frac{(1 + x)\delta}{2} \approx \frac{\delta}{2} \approx \frac{g(w)\delta}{2}.$$

Thus, with $1 + q := (1 + \epsilon_1)(1 + \epsilon_2)(1 + \epsilon_3)^{-1}(1 + \epsilon_4)$,

$$\widehat{f} = xg(\widehat{w})(1 + \theta) \approx xg(w)(1 + \theta) = f(x)(1 + \theta),$$

showing that $\widehat{f}$ is an accurate approximation to $f(x) = \log(1 + x)$.

1.7. From (1.4) we have

$$\begin{aligned}(n-1)V(x + \Delta x) &= \sum_{i=1}^{n}\Big(x_i + \Delta x_i - \overline{x} - n^{-1}\sum_{j=1}^{n}\Delta x_j\Big)^2 \\ &= (n-1)V(x) + 2\sum_{i=1}^{n}(x_i - \overline{x})\Big(\Delta x_i - n^{-1}\sum_{j=1}^{n}\Delta x_j\Big) \\ &\quad + \sum_{i=1}^{n}\Big(\Delta x_i - n^{-1}\sum_{j=1}^{n}\Delta x_j\Big)^2 \\ &= (n-1)V(x) + 2\sum_{j=1}^{n}(x_i - \overline{x})\Delta x_i \\ &\quad + \sum_{i=1}^{n}\Big(\Delta x_i - n^{-1}\sum_{j=1}^{n}\Delta x_j\Big)^2.\end{aligned}$$

This yields the following inequality, which is attainable to first order:

$$(n-1)|V(x)-V(x+\Delta x)| \le 2\epsilon \sum_{i=1}^{n} |x_i-\overline{x}||x_i| + \epsilon^2 \sum_{i=1}^{n} \Big(|x_i| + n^{-1}\sum_{j=1}^{n} |x_j|\Big)^2.$$

Dividing by $\epsilon(n-1)V(x)$ and taking the limit as $\epsilon \to 0$ gives the expression for κ_C. The result for κ_N follows from

$$\left|\sum_{j=1}^{n} (x_i-\overline{x})\Delta x_i\right| \le \left(\sum_{i=1}^{n} (x_i-\overline{x})^2\right)^{1/2} \epsilon\|x\|_2 = \epsilon(n-1)^{1/2}V(x)^{1/2}\|x\|_2,$$

where the inequality is attainable (Cauchy–Schwarz), together with the relation $(n-1)V(x) = \|x\|_2^2 - n\overline{x}^2$. That $\kappa_N \ge \kappa_C$ is easily verified.

1.8. The general solution to the recurrence is

$$x_k = \frac{100^{k+1}a + 6^{k+1}b + 5^{k+1}c}{100^k a + 6^k b + 5^k c},$$

where *a*, *b*, and *c* are arbitrary constants. The particular starting values chosen yield *a* = 0, *b* = *c* = 1, so that

$$x_k = \frac{6^{k+1}+5^{k+1}}{6^k+5^k} = 6 - \frac{1}{1+(6/5)^k}.$$

Rounding errors in the evaluation (and in the representation of x_1 on a binary machine) cause a nonzero "a" term to be introduced, and the computed values are approximately of the form

$$\widehat{x}_k \approx \frac{100^{k+1}\eta + 6^{k+1} + 5^{k+1}}{100^k\eta + 6^k + 5^k},$$

for a constant *h* of order the unit roundoff. Hence the computed iterates rapidly converge to 100. Note that resorting to higher precision merely delays the inevitable convergence to 100. Priest [844, 1992, pp. 54–56] gives some interesting observations on the stability of the evaluation of the recurrence.

1.9. Writing

$$C := \operatorname{adj}(A) = \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix},$$

we have

$$\begin{aligned} \widehat{x} &= fl\left(\frac{1}{d}Cb\right) \\ &= \frac{1}{d}(C+\Delta C)b, \qquad |\Delta C| \le \gamma_3|C|, \\ &= x + \frac{1}{d}\Delta Cb. \end{aligned}$$

so

$$\frac{\|x-\widehat{x}\|_\infty}{\|x\|_\infty} \le \gamma_3 \frac{\| |A^{-1}||b| \|_\infty}{\|x\|_\infty} \le \gamma_3 \operatorname{cond}(A,x).$$

Also, $r = b - A\widehat{x} = -(1/d)A\Delta Cb$, so $|r| \le \gamma_3 |A||A^{-1}||b|$, which implies the normwise residual bound. Note that the residual bound shows that $\|r\|_\infty/(\|A\|_\infty\|x\|_\infty)$ will be small if x is a large-normed solution $(\|x\|_\infty \approx \|A^{-1}\|_\infty\|b\|_\infty)$.

1.10. Let $\widehat{m} = fl(\overline{x})$. For any standard summation method we have (see §4.2)

$$\begin{aligned}\widehat{m} &= n^{-1}\sum_{i=1}^n x_i(1+\delta_i), \qquad |\delta_i| \le \gamma_{n-1},\\ &= m + \Delta m, \qquad \Delta m := n^{-1}\sum_{i=1}^n x_i\delta_i.\end{aligned}$$

Then, since $fl\big((x_i-\widehat{m})^2\big) = (x_i-\widehat{m})^2(1+\alpha_i)^2(1+\beta_i)$, $|\alpha_i|, |\beta_i| \le u$, $\widehat{V}$ satisfies

$$\begin{aligned}(n-1)\widehat{V} &= \sum_{i=1}^n (x_i-\widehat{m})^2(1+\epsilon_i), \qquad |\epsilon_i| \le \gamma_{n+3}\\ &= \sum_{i=1}^n \big[(x_i-m)^2 - 2\Delta m(x_i-m) + \Delta m^2\big](1+\epsilon_i)\\ &= (n-1)V + \sum_{i=1}^n (x_i-m)^2\epsilon_i\\ &\qquad + \sum_{i=1}^n (-2\Delta m(x_i-m) + \Delta m^2)(1+\epsilon_i).\end{aligned}$$

Hence, defining $\mu := n^{-1}\sum_{i=1}^n |x_i|$, so that $|\Delta m| \le \gamma_{n-1}\mu$,

$$\begin{aligned}(n-1)|V-\widehat{V}| &\le \gamma_{n+3}(n-1)V + \sum_{i=1} |-2\Delta m(x_i-m)\epsilon_i + \Delta m^2(1+\epsilon_i)|\\ &\le \gamma_{n+3}(n-1)V + 2\gamma_{n-1}\gamma_{n+3}\mu\sqrt{n(n-1)V} + \gamma_{n-1}^2\mu^2 n(1+\gamma_{n+3}),\end{aligned}$$

that is,

$$\begin{aligned}\frac{|V-\widehat{V}|}{V} &\le \gamma_{n+3} + 2\theta\gamma_{n-1}\gamma_{n+3} + \theta^2\gamma_{n-1}^2(1+\gamma_{n+3})\\ &= (n+3)u + O(u^2),\end{aligned}$$

where

$$\theta = \mu\sqrt{\frac{n}{(n-1)V}} \le \frac{\|x\|_2}{\sqrt{(n-1)V}} = \frac{1}{2}\kappa_N.$$

2.1. There are $1 + 2(e_{\max} - e_{\min} + 1)(b-1)b^{t-1}$ normalized numbers (the "1" is for zero), and $2(b^{t-1} - 1)$ subnormal numbers. For IEEE arithmetic we therefore have

	Normalized	Subnormal
single precision	4.3×10^9	1.7×10^7
double precision	1.8×10^{19}	9×10^{15}

2.2. Without loss of generality suppose $x > 0$. We can write $x = m \times b^{e-t}$, where $b^{t-1} \le m < b^t$. The next larger floating point number is $x + \Delta x$, where $\Delta x = b^{e-t}$, and

$$\beta^{-1}\epsilon_M|x| = \beta^{-t}|x| < \frac{|x|}{m} = |\Delta x| \le \beta^{e-t}(\beta^{1-t}m) = \epsilon_M|x|.$$

The same upper bound clearly holds for the "next smaller" case, and the lower bound in this case is also easy to see.

2.3. The answer is the same for all adjacent nonzero pairs of single precision numbers. Suppose the numbers are 1 and $1 + \epsilon_M$(single) $= 1 + 2^{-23}$. The spacing of the double precision numbers on [1,2] is 2^{-52}, so the answer is $2^{29} - 1 \approx 5.4 \times 10^8$.

2.4. Inspecting the proof of Theorem 2.2 we see that $|y_i| \ge b^{e-1}$, $i = 1, 2$, and so we also have $|fl(x) - x|/|fl(x)| \le u$, that is, $fl(x) = x/(1 + \delta)$, $|\delta| \le u$. Note that this is not the same δ as in Theorem 2.2, although it has the same bound, and unlike in Theorem 2.2 there can be equality in this bound for δ.

2.5. The first part is trivial. Since, in binary notation,

$$x = 0.1100\ 1100\ 1100\ 1100\ 1100\ 1100 \mid 1100 \ldots \times 2^{-3},$$

we have, rounding to 24 bits,

$$\widehat{x} = 0.1100\ 1100\ 1100\ 1100\ 1100\ 1101 \times 2^{-3}.$$

Thus

$$\begin{aligned} \widehat{x} - x &= (0.001 - 0.000\overline{1100})_2 \times 2^{-21} \times 2^{-3} \\ &= \left(\frac{1}{8} - \frac{1}{10}\right) \times 2^{-24} = \frac{1}{40} \times 2^{-24}, \end{aligned}$$

and so $(x - \widehat{x})/x = -\frac{1}{4} \times 2^{-24} = -\frac{1}{4}u$.

2.6. Since the double precision mantissa contains 53 bits, $p = 2^{53} = 9007199254740992 \approx 9.01 \times 10^{15}$. For single precision, $p = 2^{24} = 16777216 \approx 1.68 \times 10^7$.

2.7.

1. True, since in IEEE arithmetic $fl(a \text{ op } b)$ is defined to be the rounded value of a op b, round(a op b), which is the same as round(b op a).
2. True for round to nearest (the default) and round to zero, but false for round to $\pm\infty$.
3. True, because $fl(a + a) := \text{round}(a + a) = \text{round}(2 * a) =: fl(2 * a)$.
4. True: similar to 3.
5. False, in general.

6. True for binary arithmetic. Since the division by 2 is exact, the inequality is equivalent to $2a < fl(a + b) \le 2b$. But $a + b \le 2b$, so, by the monotonicity of rounding, $fl(a + b) = \text{round}(a + b) \le \text{round}(2b) = 2b$. The lower bound is verified similarly.

2.8. Examples in 3-digit decimal arithmetic are $f1((5.01 + 5.02)/2) = f1(10.0/2) = 5.0$ and $fl((5.02 + 5.04)/2) = f1(10.1/2) = 5.05$.

The left-hand inequality is immediate, since $fl((b - a)/2) \ge 0$. The right-hand inequality can possibly be violated only if a and b are close (otherwise $(a + b)/2$ is safely less than b, so the inequality will hold). But then, by Theorem 2.5, $fl(b - a)$ is obtained exactly, and the result will have several zero low-order digits in the mantissa because of cancellation. Consequently, if the base is even the division is done exactly, and the right-hand inequality follows by the monotonicity of rounding. Alternatively, and even more simply, for any base we can argue that $fl((b - a)/2) \le b - a$, so that $a + fl((b - a)/2) \le b$ and, again, the result follows by the monotonicity of rounding.

2.9.

$$\begin{aligned}\sqrt{1-2^{-53}} &= 1 - \frac{1}{2}\cdot 2^{-53} - \frac{1}{8}\cdot 2^{-106} - \cdots \\ &= 0.\underbrace{11\ldots1}_{53}0\underbrace{11\ldots1}_{54}011\ldots \quad \text{(binary).}\end{aligned}$$

Rounded directly to 53 bits this yields $1 - 2^{-53}$. But rounded first to 64 bits it yields

$$0.\underbrace{11\ldots1}_{53}1\underbrace{00\ldots0}_{10},$$

and when this number is rounded to 53 bits using the round to even rule it yields 1.0.

2.12. The spacing between the floating point numbers in the interval (1/2,1] is $\epsilon/2$ (cf. Lemma 2.1), so $|1/x - fl(1/x)| \le \epsilon/4$, which implies that $|1 - xfl(1/x)| \le \epsilon x/4 < \epsilon/2$. Thus

$$1 - \epsilon/2 < xfl(1/x) < 1 + \epsilon/2.$$

Since the spacing of the floating point numbers just to the right of 1 is ϵ, $xfl(1/x)$ must round to either $1 - \epsilon/2$ or 1.

2.13. If there is no double rounding the answer is 257,736,490. For a proof that combines mathematical analysis with computer searching, see Edelman [343, 1994].

2.15. The IEEE standard does not define the results of exponentiation. The choice 0° = 1 can be justified in several ways. For example, if $p(x) = \sum_{j=0}^{n} a_j x^j$ then $p(0) = a_0 = a_0 \text{ x } 0^0$, and the binomial expansion $(x + y)^n = \sum_{k=0}^{n} \binom{n}{k} x^k y^{n-k}$ yields $1 = 0^0$ for $x = 0$, $y = 1$. For more detailed discussions see Goldberg [457, 1991, p. 32] and Knuth [669, 1992, pp. 406–408].

2.17. For IEEE arithmetic the answer is no. Since $fl(x \text{ op } y)$ is defined to be the rounded version of x op y, $b^2 - ac \geq 0$ implies $fl(b^2) - fl(ac) \geq 0$ (rounding is a monotonic operation). The final computed answer is

$$\begin{aligned} fl(fl(b^2) - fl(ac)) &= (fl(b^2) - fl(ac))(1+\delta), \qquad |\delta| \leq u \\ &\geq 0. \end{aligned}$$

2.18. No. A counterexample in 2-digit base-10 arithmetic is $fl(2.0 - 0.91) = fl(1.09) = 1.1$.

2.19. The function $fl(\sqrt{x})$ maps the set of positive floating point numbers onto a set of floating point numbers with about half as many elements. Hence there exist two distinct floating point numbers x having the same value of $fl(\sqrt{x})$, and so the condition $(\sqrt{x})^2 = |x|$ cannot always be satisfied in floating point arithmetic. The requirement $\sqrt{x^2} = |x|$ is reasonable for base 2, however, and is satisfied in IEEE arithmetic, as we now show.

Without loss of generality, we can assume that $1 \leq x < 2$, since scaling x by a power of 2 does not alter $fl(\sqrt{x^2})$. By definition, $fl(\sqrt{x^2})$ is the nearest floating point number to $\sqrt{fl(x^2)} = \sqrt{x^2(1+\delta)}$, $|\delta| \leq u$, and

$$\sqrt{fl(x^2)} = |x|(1+\delta)^{1/2} = |x|\left(1 + \frac{\delta}{\sqrt{1+\delta}+1}\right) = |x| + \theta.$$

Now the spacing of the floating point numbers between 1 and 2 is $\epsilon_M = 2u$, so

$$|\theta| \leq (2-2u)\frac{u}{\sqrt{1-u}+1}.$$

Hence $|\theta| < u$ if $u \leq 1/4$ (say), and then $|x|$ is the nearest floating point number to $\sqrt{fl(x^2)}$, so that $fl(\sqrt{fl(x^2)}) = |x|$.

In base-10 floating point arithmetic, the condition $\sqrt{x^2} = |x|$ can be violated. For example, working to 5 significant decimal digits, if $x = 3.1625$ then $fl(x^2) = fl(10.0014\,0625) = 10.001$, and $fl(\sqrt{10.001}) = fl(3.1624\,3577\ldots) = 3.1624 < x$.

2.20. On a Cray Y-MP the answer is yes, but in base-2 IEEE arithmetic the answer is no. It suffices to demonstrate that $fl(x/\sqrt{x^2}) = \text{sign}(x)$ $(x \neq 0)$, which is shown by the proof of Problem 2.19.

2.21. The test "$x > y$" returns false if x or y is a NaN, so the code computes max(NaN, 5) = 5 and max(5, NaN) = NaN, which violates the obvious requirement that $\max(x, y) = \max(y, x)$. Since the test $x \neq x$ identifies a NaN, the following code implements a reasonable definition of $\max(x, y)$:

```
% max($x$, $y$)
if $x \neq x$ then
  max = $y$
else if $y \neq y$ then
      max = $x$
   else if $y > x$ then
```

```
              max = y
            else
              max = x
            end
        end
    end
```

A further refinement is to ensure that max(-0, + 0) = +0, which is not satisfied by the code above since –0 and +0 compare as equal; this requires bit-level programming.

2.22. We give an informal proof; the details are obtained by using the model $fl(x \text{ op } y) = (x \text{ op } y)(1+\delta)$, but they obscure the argument.

Since a, b, and c are nonnegative, $a + (b + c)$ is computed accurately. Since $c \le b \le a$, $c + (a - b)$ and $a + (b - c)$ are the sums of two positive numbers and so are computed accurately. Since a, b, and c are the lengths of sides of a triangle, $a \le b + c$; hence, using $c \le b \le a$,

$$b \le a \le b + c \le 2b,$$

which implies that $a - b$ is computed exactly, by Theorem 2.5. Hence $c - (a - b)$ is the difference of two exactly represented numbers and so is computed accurately. Thus $fl(A) = fl(\frac{1}{4}\sqrt{\widehat{x}})$, where $\widehat{x}$ is an accurate approximation to the desired argument x of the square root. It follows that $fl(A)$ is accurate.

2.23. For a machine with a guard digit, $y = x$, by Theorem 2.5 (assuming $2x$ does not overflow). If the machine lacks a guard digit then the subtraction produces x if the last bit of x is zero, otherwise it produces an adjacent floating point number with a zero last bit; in either case the result has a zero last bit. Gu, Demmel, and Dhillon [484, 1994] apply this bit zeroing technique to numbers $d_1, d_2, \ldots, d_n$ arising in a divide and conquer bidiagonal SVD algorithm, their motivation being that the differences $d_i - d_j$ can then be computed accurately even on machines without a guard digit.

2.24. The function $f(x) = 3x - 1$ has the single root $x = 1/3$. We can have $fl(f(x)) = 0$ only for $x \approx 1/3$. For $x \approx 1/3$ the evaluation can be summarized as follows:

$$\begin{aligned} x - 0.5 &\approx -\frac{1}{6}, \\ -\frac{1}{6} + x &\approx \frac{1}{6}, \\ \frac{1}{6} - 0.5 &\approx -\frac{1}{3}, \\ -\frac{1}{3} + x &\approx 0. \end{aligned}$$

The first, second, and fourth subtractions are done exactly, by Theorem 2.5. The result of the first subtraction has a zero least-significant bit and the result of the second has two zero least-significant bits; consequently the third subtraction suffers no loss of trailing bits and is done exactly, Therefore $f(x)$ is computed exactly for $x \equiv fl(x)$ near 1/3. But $fl(x)$ can never equal 1/3 on a binary machine, so $fl(f(x)) \neq 0$ for all x.

2.25. We have

$$\begin{aligned} \widehat{w} &= bc(1+\delta_1), \\ \widehat{e} &= (\widehat{w} - bc)(1+\delta_2), \\ (1+\delta_4)\widehat{x} &= (ad - \widehat{w})(1+\delta_3) + \widehat{e}, \end{aligned}$$

where $|\delta_i| \le u$, $i = 1{:}4$. Hence

$$(1+\delta_4)\widehat{x} = (ad - bc - bc\delta_1)(1+\delta_3) + bc\delta_1(1+\delta_2) = x + x\delta_3 - bc\delta_1(\delta_3 - \delta_2),$$

so that

$$|x - \widehat{x}| \le u(|x| + |\widehat{x}|) + 2u^2|bc|,$$

which implies high relative accuracy unless $u|bc| >> |x|$. For comparison, the bound for standard evaluation of $fl(ad - bc)$ is $|x - \widehat{x}| \le \gamma_2(|ad| + |bc|)$.

2.26. Newton's method is

$$\begin{aligned} x_{k+1} &= x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{a - 1/x_k}{x_k^{-2}} \\ &= x_k + (1 - x_k a)x_k. \end{aligned}$$

The quadratic convergence can be seen from $1 - x_{k+1}a = (1 - x_k a)^2$.

2.27. We would like (2.8) to be satisfied, that is, we want $\widehat{z} := fl(x/y)$ to satisfy

$$\widehat{z} = (x/y)(1+\delta) + \eta, \qquad |\delta| \le u, \quad |\eta| \le \lambda u. \tag{A.1}$$

This implies that

$$|\widehat{z}y - x| = |x\delta + \eta y| \le u|x| + \lambda u|y|.$$

In place of $r = \widehat{z}y - x$ we have to use $\widehat{r} = fl(\widehat{z}y - x) = \widehat{z}y(1+\epsilon) - x$, where $|\epsilon| \le u$, and where we can assume the subtraction is exact, since the case of interest is where $\widehat{z}y \approx x$ (see Theorem 2.5). Thus $|\widehat{r}| = |r + \widehat{z}y\epsilon| \le |r| + u|\widehat{z}||y|$, and bounding $|\widehat{z}||y|$ using (Al) we obtain

$$|\widehat{r}| \le |r| + u(1+u)|x| + \lambda u^2|y| \approx |r| + u|x|.$$

Therefore the convergence test is

$$|\widehat{r}| \le 2u|x| + \lambda u|y|.$$

Since λu underflows to zero it cannot be precomputed, and we should instead compute $\lambda(u|y|)$.

3.1. The proof is by induction. Inductive step: for $r_n = +1$,

$$\begin{aligned}\prod_{i=1}^{n}(1+\delta_i) &= (1+\delta_n)(1+\theta_{n-1}) = 1+\theta_n,\\ \theta_n &= \delta_n + (1+\delta_n)\theta_{n-1},\\ |\theta_n| &\le u + (1+u)\frac{(n-1)u}{1-(n-1)u}\\ &= \frac{u(1-(n-1)u)+(1+u)(n-1)u}{1-(n-1)u}\\ &= \frac{nu}{1-(n-1)u} \le \gamma_n.\end{aligned}$$

For $r_n = -1$ we find, similarly, that

$$|\theta_n| \le \frac{nu-(n-1)u^2}{1-nu+(n-1)u^2} \le \gamma_n.$$

3.2. This result can be proved by a modification of the proof of Lemma 3.1. But it follows immediately from the penultimate line of the proof of Lemma 3.4.

3.3. The computed iterates $\widehat{q}_k$ satisfy

$$(1+\delta_k)\widehat{q}_k = a_k + \frac{b_k}{\widehat{q}_{k+1}}(1+\epsilon_k), \qquad |\delta_k|, |\epsilon_k| \le u.$$

Defining $\widehat{q}_k = q_k + e_k$, we have

$$\begin{aligned}q_k + e_k + \delta_k\widehat{q}_k &= a_k + \frac{b_k}{q_{k+1}+e_{k+1}} + \frac{b_k\epsilon_k}{\widehat{q}_{k+1}}\\ &= a_k + \frac{b_k}{q_{k+1}} - \frac{b_k e_{k+1}}{q_{k+1}(q_{k+1}+e_{k+1})} + \frac{b_k\epsilon_k}{\widehat{q}_{k+1}}.\end{aligned}$$

This gives

$$|e_k| \le u|\widehat{q}_k| + \frac{|b_k||e_{k+1}|}{(|\widehat{q}_{k+1}| - |e_{k+1}|)|\widehat{q}_{k+1}|} + u\frac{|b_k|}{|\widehat{q}_{k+1}|}.$$

The running error bound m can therefore be computed along with the continued fraction as follows:

```
$q_{n+1} = a_{n+1}$
$f_{n+1} = 0$
for $k = n: -1:0$
    $r_k = b_k/q_{k+1}$
    $q_k = a_k + r_k$
    $f_k = |q_k| + |r_k| + |b_k|f_{k+1}/((|q_{k+1}| - uf_{k+1})|q_{k+1}|)$
end
$m = uf_0$
```

The error bound is valid provided that $|q_{k+1}| - uf_{k+1} > 0$ for all k. Otherwise a more sophisticated approach is necessary (for example, to handle the case where $q_{k+1} = 0$, $q_k = \infty$ and q_{k-1} is finite).

3.4. We prove just the division result and the last result. Let $(1+\theta_k)/(1+\theta_j) = 1+\alpha$. Then $\alpha = (\theta_k - \theta_j)/(1+\theta_j)$, so

$$|\alpha| \le \frac{\frac{ku}{1-ku} + \frac{ju}{1-ju}}{1 - \frac{ju}{1-ju}} = \frac{(k+j)u - 2kju^2}{(1-2ju)(1-ku)}$$
$$= \frac{(k+j)u - 2kju^2}{1-(2j+k)u + 2jku^2} \le \frac{(k+j)u}{1-(2j+k)u} \le \gamma_{k+2j}.$$

However, it is easy to verify that, in fact,

$$\frac{(k+j)u - 2kju^2}{(1-2ju)(1-ku)} \le \frac{(k+j)u}{1-(k+j)u}$$

if $j \le k$.

For the last result,

$$\gamma_k + \gamma_j + \gamma_k\gamma_j = \frac{ku(1-ju) + ju(1-ku) + kju^2}{(1-ku)(1-ju)}$$
$$= \frac{(k+j)u - kju^2}{1-(k+j)u + kju^2}$$
$$< \frac{(k+j)u}{1-(k+j)u} = \gamma_{k+j}.$$

3.5. We have $fl(AB) = AB + \Delta C = (A + \Delta C B^{-1})B =: (A + \Delta A)B$, where $\Delta A = \Delta C B^{-1}$, and $|\Delta C| \le \gamma_n |A||B|$ by (3.12). Similarly, $fl(AB) = A(B + \Delta B)$, $|\Delta B| \le \gamma_n |A^{-1}||A||B|$.

3.6. We have

$$|R| = |A\Delta B + \Delta AB + \Delta A \Delta B| \le 2\epsilon EF + \epsilon^2 EF,$$

which implies

$$\epsilon^2 g_{ij} + 2\epsilon g_{ij} - |r_{ij}| \ge 0 \quad \text{for all } i,j.$$

Solving these inequalities for ϵ gives the required lower bound.

The definition is flawed in general as can be seen from rank considerations. For example if A and B are vectors and $\operatorname{rank}(C) \ne 1$, then no perturbations AA and ΔB exist to satisfy $C = (A + \Delta A)(B + \Delta B)$! Thus we have to use a mixed forward/backward stability definition in which we perturb C by at most ϵ as well as A and B.

3.7. Lemma 3.5 show that (3.2) holds provided that each $(1+\delta)^k$ product is replaced by $(1+\epsilon)(1+\delta)^{k-1}$, where $|\epsilon| \le \sqrt{2}\gamma_2$, where the $1+\epsilon$ corresponds to a multiplication $x_i y_i$. Thus we have

$$\widehat{S}_n = \sum_{i=1}^{n} x_i y_i (1+\alpha_i), \quad \text{where} \quad 1+\alpha_i = (1+\theta_{n-1}^{(i)})(1+\epsilon_i).$$

It is easy to show that $|\alpha_i| \le \gamma_{n+2}$, so the only change required in (3.4) is to replace γ_n by γ_{n+2}. The complex analogue of (3.10) is $\widehat{y} = (A + \Delta A)x$, $|\Delta A| \le \gamma_{n+2}|A|$.

3.8. Without loss of generality we can suppose that the columns of the product are computed one at a time. With $xj = A_1 \,.\,.\,.\, A_k ej$ we have, using (3.10),

$$fl(x_j) = (A_1 + \Delta A_1)\dots(A_k + \Delta A_k)e_j, \qquad \|\Delta A_i\|_2 \le \sqrt{n}\gamma_n\|A_i\|_2,$$

and so, by Lemma 3.6,

$$\|x_j - fl(x_j)\|_2 \le \left[(1+\sqrt{n}\gamma_n)^k - 1\right]\|A_1\|_2\dots\|A_k\|_2.$$

Squaring these inequalities and summing over j yields

$$||A_1 \,.\,.\,.\, A_k - fl(A_l \,.\,.\,.\, A_k)||_F \le \sqrt{n}\left[(1+\sqrt{n}\gamma_n)^k - 1\right]\|A_1\|_2\dots\|A_k\|_2,$$

which gives the result.

Note that the product $||A_1||_2 \,.\,.\,.\, ||A_k||_2$ can be much smaller than $||A||_F \,.\,.\,.\, ||A_k||_F$; the extreme case occurs when the A_i are orthogonal.

3.9. We have $fl((x + y)(x - y)) = (x + y)(x - y)(1 + \theta_3)$, $|\theta_3| \le \gamma_3 = 3u/(1 - 3u)$, so the computed result has small relative error. Moreover, if $y/2 \le x \le y$ then $x - y$ is computed exactly, by Theorem 2.5, hence $fl((x + y)(x - y)) = (x + y)(x - y)\,(1 + \theta_2)$. However, $fl(x^2 - y^2) = x^2(1 + \theta_2) - y^2(1 + q'_2)$, so that

$$\left|\frac{fl(x^2-y^2)}{x^2-y^2}\right| = \left|\frac{x^2\theta_2 - y^2\theta_2'}{x^2-y^2}\right| \le \gamma_2\frac{x^2+y^2}{|x^2-y^2|},$$

and we cannot guarantee a small relative error.

If $|x| >> |y|$ then $fl(x^2 - y^2)$ suffers only two rounding errors, since the error in forming $f1(y^2)$ will not affect the final result, while $fl((x + y)(x - y))$ suffers three rounding errors; in this case $fl(x^2 - y^2)$ is likely to be the more accurate result.

3.10. Assume the result is true for $m - 1$. Now

$$\prod_{j=0}^{m}(X_j + \Delta X_j) = (X_m + \Delta X_m)\prod_{j=0}^{m-1}(X_j + \Delta X_j),$$

so

$$\begin{aligned}
\left\|\prod_{j=0}^{m}(X_j + \Delta X_j) - \prod_{j=0}^{m}X_j\right\| &= \left\|X_m\left[\prod_{j=0}^{m-1}(X_j + \Delta X_j) - \prod_{j=0}^{m-1}X_j\right]\right.\\
&\qquad\left. + \Delta X_m\prod_{j=0}^{m-1}(X_j + \Delta X_j)\right\|\\
&\le \|X_m\|\left(\prod_{j=0}^{m-1}(1+\delta_j) - 1\right)\prod_{j=0}^{m-1}\|X_j\|\\
&\qquad + \delta_m\|X_m\|_2\prod_{j=0}^{m-1}\left(\|X_j\|(1+\delta_j)\right)\\
&= \left(\prod_{j=0}^{m}(1+\delta_j) - 1\right)\prod_{j=0}^{m}\|X_j\|.
\end{aligned}$$

3.11. The computations can be expressed as

$$\begin{aligned} y_1 &= x^2 \\ y_{i+1} &= \sqrt{y_i},\ i = 1\colon m \\ z_1 &= y_{m+1} \\ z_{i+1} &= z_i^2,\ i = 1\colon m-1 \end{aligned}$$

We have

$$\begin{aligned} \widehat{y}_1 &= x^2(1+\delta_0) \\ \widehat{y}_{i+1} &= \sqrt{\widehat{y}_i}(1+\delta_i),\ i = 1\colon m \\ \widehat{z}_1 &= \widehat{y}_{m+1} \\ \widehat{z}_{i+1} &= \widehat{z}_i^{\,2}(1+\epsilon_i),\ i = 1\colon m-1 \end{aligned}$$

where $|\delta_i|$, $|\epsilon_i| \le \le u$. Solving these recurrences, we find that

$$\widehat{y}_{m+1} = (1+\delta_m)(1+\delta_{m-1})^{1/2}(1+\delta_{m-2})^{1/4}\dots(1+\delta_0)^{1/2^m}y_1^{1/2^m}.$$

It follows that

$$(1-u)^2 y_1^{\frac{1}{2^m}} \le \widehat{y}_{m+1} \le (1+u)^2 y_1^{\frac{1}{2^m}},$$

which shows that $\widehat{y}_{m+1}$ differs negligibly from y_{m+1}. For the repeated squarings, however, we find that

$$\begin{aligned} \widehat{z}_m &= (1+\epsilon_{m-1})(1+\epsilon_{m-2})^2(1+\epsilon_{m-3})^4\dots(1+\epsilon_1)^{2^{m-1}}\widehat{z}_1^{\,2^{m-1}} \\ &= (1+\theta_{2^m-1})\widehat{z}_1^{\,2^{m-1}} = (1+\theta_{2^m+1})|x|, \end{aligned}$$

where we have used Lemma 3.1. Hence the squarings introduce a relative error that can be approximately as large as $2^m u$. Since $u = 2^{-53}$ this relative error is of order 0.1 for $m = 50$, which explains the observed results for $m = 50$.

For $m = 75$, the behaviour on the Sun is analogous to that on the HP calculator described in §1.12.2. On the 486DX, however, numbers less than 1 are mapped to 1. The difference is due to the fact that the 486DX uses double rounding and the Sun does not; see Problem 2.9.

3.12. The analysis is just a slight extension of that for an inner product. The analogue of (3.3) is

$$\begin{aligned} \widehat{J}(f) = {} & w_1 f(x_1)(1+\eta_1)(1+\theta_n) + w_2 f(x_2)(1+\eta_2)(1+\theta'_n) \\ & + w_3 f(x_3)(1+\eta_3)(1+\theta_{n-1}) + \cdots + w_n f(x_n)(1+\eta_n)(1+\theta_2). \end{aligned}$$

Hence

$$|I(f) - \widehat{J}(f)| \le (\eta + \gamma_n + \eta\gamma_n)\sum_{i=1}^{n}|w_i||f(x_i)|.$$

Setting $M = \max\{\,|f(x)| : a \le x \le b\}$, we have

$$|I(f) - \widehat{J}(f)| \le M(\eta + \gamma_n + \eta\gamma_n)\sum_{i=1}^{n}|w_i|. \tag{A.2}$$

Any reasonable quadrature rule designed for polynomic f has $\sum_{i=1}^{n} w_i = b - a$, so one implication of (A.2) is that it is best not to have weights of large magnitude and varying sign; ideally, $w_i \geq 0$ for all i (as for Gaussian integration rules, for example), so that $\sum_{i=1}^{n} |w_i| = b - a$.

4.1. A condition number is

$$C(x) = \max\left\{ \left| \frac{S_n(x) - S_n(x + \Delta x)}{\epsilon S_n(x)} \right| : |\Delta x| \leq \epsilon |x| \right\}.$$

It is easy to show that

$$C(x) = \frac{S_n(|x|)}{|S_n(x)|} = \frac{\sum_{i=1}^{n} |x_i|}{\left|\sum_{i=1}^{n} x_i\right|}.$$

The condition number is 1 if the x_i all have the same sign.

4.2. In the $(i-1)$st floating point addition the "2^{k-t}" portion of x_i does not propagate into the sum (assuming that the floating point arithmetic uses round to nearest with ties broken by rounding to an even last bit or rounding away from zero), thus there is an error of 2^{k-t} and $\widehat{S}_i = i$. The total error is

$$2^{-t}(1 + 2^2 + 2^4 + \cdots + 2^{2(r-1)}) = 2^{-t}\frac{4^r - 1}{3},$$

while the upper bound of (4.4) is

$$(n-1)u\sum_{i=1}^{n} |x_i| + O(u^2) \leq 2^r 2^{-t} 2^r + O(2^{-2t}) \approx 2^{-t} 4^r,$$

which agrees with the actual error to within a factor 3; thus the smaller upper bound of (4.3) is also correct to within this factor. The example just quoted is, of course, a very special one, and as Wilkinson [1088, 1963, p. 20] explains, "in order to approach the upper bound as closely as this, not only must each error take its maximum value, but all the terms must be almost equal."

4.3. With $S_k = \sum_{i=1}^{k} x_i$, we have

$$\widehat{S}_k = fl(\widehat{S}_{k-1} + x_k) = (\widehat{S}_{k-1} + x_k)(1 + \delta_k), \qquad |\delta_k| \leq u, \quad k = 2{:}n.$$

By repeated use of this relation it follows that

$$\widehat{S}_n = (x_1 + x_2)\prod_{k=2}^{n}(1 + \delta_k) + \sum_{i=3}^{n} x_i \prod_{k=i}^{n}(1 + \delta_k),$$

which yields the required expression for $\widehat{S}_n$. The bound on $|E_n|$ is immediate.

The bound is minimized if the x_i are in increasing order of absolute value. This observation is common in the literature and it is sometimes used to conclude that the increasing ordering is the best one to use. This reasoning is fallacious, because" minimizing an error bound is not the same as minimizing the error itself. As (4.3) shows, if we know nothing about the signs of the rounding errors then the "best" ordering to choose is one that minimizes the partial sums.

4.4. Any integer between 0 and 10 inclusive can reproduced. For example, $fl(1 + 2 + 3 + 4 + M - M) = 0$, $fl(M - M + 1 + 2 + 3 + 4) = 10$, and $fl(2 + 3 + M - M + 1 + 4) = 5$.

4.5. This method is sometimes promoted on the basis of the argument that it minimizes the amount of cancellation in the computation of S_n. This is incorrect: the "±" method does not reduce the amount of cancellation—it simply concentrates all the cancellation into one step. Moreover, cancellation is not a bad thing per se, as explained in §1.7.

The "±' method is an instance of Algorithm 4.1 (assuming that S_+ and S_- are computed using Algorithm 4.1) and it is easy to see that it maximizes $\max_i |T_i|$ over all methods of this form (where, as in §4.2, T_i is the sum computed at the ith stage). Moreover, when $\sum_{i=1}^{n} |x_i| \gg |\sum_{i=1}^{n} x_i|$ the value of $\max_i |T_i|$ tends to be much larger for the "±" method than for the other methods we have considered.

4.6. The main concern is to evaluate the denominator accurately when the x_i are close to convergence. The bound (4.3) tells us to minimize the partial sums; these are, approximately, for $x_i \approx \xi$, (a) $-\xi$, 0, (b) 0, 0, (c) 2ξ, 0. Hence the error analysis of summation suggests that (b) is the best expression, with (a) a distant second. That (b) is the best choice is confirmed by Theorem 2.5, which shows there will be only one rounding error when the x_i are close to convergence. A further reason to prefer (b) is that it is less prone to overflow than (a) and (c).

4.7. This is, of course, not a practical method, not least because it is very prone to overflow and underflow. However, its error analysis is interesting. Ignoring the error in the log evaluation, and assuming that exp is evaluated with relative error bounded by u, we have, with $|\delta| \le u$ for all i, and for some δ_{2n}

$$\begin{aligned}\widehat{S}_n &= \log\big(e^{x_1}\dots e^{x_n}(1+\delta_1)\dots(1+\delta_{2n-1})\big)\\ &= \log\big(e^{x_1}\dots e^{x_n}(1+\delta_{2n})^{2n-1}\big)\\ &= x_1 + \dots + x_n + (2n-1)\log(1+\delta_{2n})\\ &\approx x_1 + \dots + x_n + (2n-1)\delta_{2n}.\end{aligned}$$

Hence the best relative error bound we can obtain is $|S_n - \widehat{S}_n|/|S_n| \lesssim (2n-1)u/|S_n|$. Clearly, this method of evaluation guarantees a small absolute error, but not a small relative error when $|S_n| << 1$.

4.8. Method (a) is recursive summation of a, h, h,. . . . h. From (4.3) we have $|a + \widehat{ih} - \widehat{x}_i| \le [ia + \frac{1}{2}i(i+1)\widehat{h}]u + O(u^2)$. Hence, since $h = h(1+\epsilon)$, $|\epsilon| \le u$,

$$\begin{aligned}|x_i - \widehat{x}_i| &\le |ih - \widehat{ih}| + |a + \widehat{ih} - \widehat{x}_i| \le ihu + i(|a| + \tfrac{1}{2}(i+1)h)u + O(u^2)\\ &= i(|a| + \tfrac{1}{2}(i+3)h)u + O(u^2).\end{aligned}$$

For (b), using the relative error counter notation (3.9), $\widehat{x}_i = (a + \widehat{ih}\langle 1\rangle)\langle 1\rangle = a\langle 1\rangle + ih\langle 3\rangle$. Hence

$$|x_i - \widehat{x}_i| \le u|a| + \gamma_3 ih.$$

For (c), $\widehat{x}_i = a(1 - \langle 1\rangle\, i/n)\langle 2\rangle + (i/n)\, b\langle 3\rangle$, hence

$$|x_i - \widehat{x}_i| \le \gamma_2|a| + \gamma_3|a|i/n + \gamma_3|b|i/n \le \gamma_3\big(|a|(1+i/n) + |b|i/n\big).$$

The error bound for (b) is about a factor i smaller than that for (a). Note that method (c) is the only one guaranteed to yield $\widehat{x}_n = b$ (assuming $fl(n/n) = 1$, as holds in IEEE arithmetic), which may be important when integrating a differential equation to a given end-point.

If $a \ge 0$ then the bounds imply

$$\begin{aligned}
(a):&\quad |x_i - \widehat{x}_i| \lesssim iu|x_i|,\\
(b):&\quad |x_i - \widehat{x}_i| \le \gamma_3|x_i|,\\
(c):&\quad |x_i - \widehat{x}_i| \le \gamma_3(a + i/n(a+b)) = \gamma_3(a + i/n(b - a + 2a))\\
&\qquad\qquad\qquad = \gamma_3(a + ih + 2ai/n) \le 3\gamma_3|x_i|.
\end{aligned}$$

Thus (b) and (c) provide high relative accuracy for all i, while the relative accuracy of (a) can be expected to degrade as i increases.

5.1. By differentiating the Homer recurrence $q_i = xq_{i+1} + a_i$, $q_n = a_n$, we obtain

$$\begin{aligned}
q_i' &= xq_{i+1}' + q_{i+1}, & q_n' &= 0, & q_{n-1}' &= a_n,\\
q_i'' &= xq_{i+1}'' + 2q_{i+1}', & q_n'' &= q_{n-1}'' = 0, & q_{n-2}'' &= 2a_n,\\
q_i''' &= xq_{i+1}''' + 3q_{i+1}'', & q_n''' &= q_{n-1}''' = q_{n-2}''' = 0, & q_{n-3}''' &= 6a_n.
\end{aligned}$$

The factors 2, 3, ... can be removed by redefining $r_i^{(k)} = q_i^{(k)}/k!$. Then $r_i^{(k)} = xr_{i+1}^{(k)} + r_{i+1}^{(k-1)}$, $r_{n-k}^{(k)} = a_n$.

5.2. Analysis similar to that for Homer's rule shows that

$$fl(p(x)) = a_0\langle n\rangle + a_1x\langle n+1\rangle + \cdots + a_nx^n\langle n+1\rangle.$$

The total number of rounding errors is the same as for Homer's algorithm, but they are distributed more equally among the terms of the polynomial. Homer's rule can be expected to be more accurate when the terms $|a_ix^i|$ decrease rapidly with i, such as when $p(x)$ is the truncation of a rapidly convergent power series. Of course, this algorithm requires twice as many multiplications as Homer's method.

5.3. Accounting for the error in forming y, we have, using the relative error counter notation (3.9),

$$\begin{aligned}
fl(p(x)) &= \big[a_0\langle 1\rangle + a_2x^2\langle 4\rangle + \cdots + a_{n-2}x^{n-2}\langle 3m-2\rangle + a_nx^n\langle 3m\rangle\\
&\qquad + x\langle 1\rangle\big(a_1\langle 1\rangle + a_3x^2\langle 4\rangle + \cdots + a_{2n-1}x^{2n-1}\langle 3m-3\rangle\big)\big]\langle 1\rangle\\
&= \sum^{n} a_ix^i(1+\theta_{3m+1}^{(i)}), \qquad |\theta_{3m+1}^{(i)}| \le \gamma_{3m+1}.
\end{aligned}$$

Thus the relative backward perturbations are bounded by $(3n/2 + 1)u$ instead of $2nu$ for Homer's rule.

5.4. Here is a MATLAB M-file to perform the task.

```
function [a, perm] = leja(a)
%LEJA    LEJA  ordering.
%         [A, PERM] = LEJA(A) reorders the points A by the
%        Leja ordering and returns the permutation vector that
%        effects the ordering in PERM.

n  =  max(size(a));
perm = (l:n)';

% a(1)  = max(abs(a)).
[t,  i]  = max(abs(a));
if i ~= 1
   a([1 i]) = a([i 11);
   perm([1 i]) = perm([i l]);
end

p  =  ones(n,l);
for k = 2 : n - 1
    for i = k:n
       p(i)  =  p(i)*(a(i)-a(k-1))  ;
    end
    [t,  il  =  max(abs(p(k:n)));
    i = i+k-l;
    if i ~= k
       a([k i]) = a([i k]);
       p([k  i])  = p([ik]);
       perm([k i]) = perm([i k]);
    end
end
```

5.5. It is easy to show that the computed $\widehat{p}$ satisfies $\widehat{p} = p(x)(1 + q_{2n+1})$, $|q_{2n+1}| \le \gamma_{2n+1}$. Thus $\widehat{p}$ has a tiny relative error. Of course, this assumes that the roots x_i are known exactly!

6.1. For $\alpha_{S,2}$, if $A \in \mathbb{C}^{m\times n}$ then, using the Cauchy–Schwarz inequality,

$$\|A\|_S^2 = \Big(\sum_{i,j} |a_{ij}|\Big)^2 \le mn\|A\|_F^2 \le mn\,\mathrm{rank}(A)\|A\|_2^2.$$

The first inequality is an equality iff $|a_{ij}| \equiv \alpha$, and the second inequality is an equality iff A is a multiple of a matrix with orthonormal columns. If A is real and square, these requirements are equivalent to A being a scalar multiple of a Hadamard matrix. If A is complex and square, the requirements are satisfied by the given Vandermonde matrix, which is $\sqrt{n}$ times a unitary matrix.

6.2. $\|xy^*\| = \max_{\|z\|=1} \|xy^*z\| = \|x\| \max_{\|z\|=1} |y^*z| = \|x\|\,\|y\|_D$.

6.3. By the Holder inequality,

$$\operatorname{Re} y^*Ax \le |y^*Ax| \le \|y\|_D\|A\|\,\|x\|. \tag{A.3}$$

We now show that equality is possible throughout (A.3). Let x satisfy $\|A\| = \|Ax\|/\|x\|$ and let y be dual to Ax. Then

$$\operatorname{Re} y^*Ax = y^*Ax = \|y\|_D\|Ax\| = \|y\|_D\|A\|\,\|x\|,$$

as required.

6.4. From (6.19) we have $\|M_n\|_p \le \mu_n^{1/p}\mu_n^{1-1/p} = \mu_n$. But by taking x in the definition (6. 11) to be the vector of all ones, we see that $\|M_n\|_P \ge m_n$.

6.5. If $A = PDQ^*$ is an SVD then

$$\begin{aligned}\|AB\|_F &= \|PDQ^*B\|_F = \|DQ^*B\|_F\\ &\le \|\operatorname{diag}(\max_i \sigma_i)Q^*B\|_F = (\max_i \sigma_i)\|Q^*B\|_F\\ &= \|A\|_2\|B\|_F.\end{aligned}$$

Similarly, $\|BC\|_F \le \|B\|_F\|C\|_2$, and these two inequalities together imply the required one.

6.6. By (6.6) and (6.8) it suffices to show that $\|A^{-1}\|_{\beta,\alpha} = (\min_{x\neq0}\|Ax\|_\beta/\|x\|_\alpha)^{-1}$. We have

$$\begin{aligned}\|A^{-1}\|_{\beta,\alpha} &= \max_{x\neq0}\frac{\|A^{-1}x\|_\alpha}{\|x\|_\beta}\\ &= \max_{y\neq0}\frac{\|y\|_\alpha}{\|Ay\|_\beta} \quad (y = A^{-1}x)\\ &= (\min_{x\neq0}\|Ax\|_\beta/\|x\|_\alpha)^{-1}.\end{aligned}$$

6.7. Let λ be an eigenvalue of A and x the corresponding eigenvector, and form the matrix $X = [x, x,. . . ,x] \in \mathbb{C}^{n\times n}$. Then $AX = \lambda X$, so $|\lambda|\|X\| = \|AX\| \le \|A\|\,\|X\|$, showing that $|\lambda| \le \|A\|$. For a subordinate norm it suffices to take norms in the equation $Ax = \lambda x$.

6.8. The following proof is much simpler than the usual proof based on diagonal scaling to make the off-diagonal of the Jordan matrix small (see, e.g., Horn and Johnson [580, 1985, Lem. 5.6.10]). The proof is from Ostrowski [812, 1973, Thin. 19.3].

Let $\delta^{-1} A$ have the Jordan canonical form $\delta^{-1} A = XJX^{-1}$. We can write $J = \delta^{-1}D + N$, where $D = \operatorname{diag}(\lambda_i)$ and the λ_i are the eigenvalues of A. Then $A = X(D + dN)X^{-1}$, so

$$\|A\| := \|X^{-1}AX\|_\infty = \|D + \delta N\|_\infty \le \rho(A) + \delta.$$

Note that we actually have $\|A\| = r(A) + d$ if the largest eigenvalue occurs in a Jordan block of size greater than 1. If A is diagonalizable then with $d = 0$ we get $\|A\| = r(A)$. The last part of the result is trivial.

6.9. Let A have the SVD $A = U\Sigma V^*$. By the unitary invariance of the 2- and Frobenius norms, $\|A\|_2 = \|\Sigma\|_2 = \sigma_1$, $\|A\|_F = \|\Sigma\|_F = \left(\sum_{i=1}^n \sigma_i^2\right)^{1/2}$. Thus $\|A\|_2 \le \|A\|_F \le \sqrt{n}\|A\|_2$ (in fact, we can replace $\sqrt{n}$ by $\sqrt{r}$, where $r = \operatorname{rank}(A)$). There is equality on the left when $\sigma_2 = \cdots = \sigma_n = 0$, that is, when A has rank 1 ($A = xy^*$) or $A = 0$. There is equality on the right when $\sigma_1 = \cdots = \sigma_n = \alpha$, that is, when $A = \alpha Q$ where Q has orthonormal columns, $\alpha \in \mathbb{C}$.

6.10. Let $F = P\Sigma Q^*$ be an SVD. Then

$$\left\|\begin{bmatrix} I & F \\ 0 & I \end{bmatrix}\right\|_2 = \left\|\begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix}\begin{bmatrix} I & \Sigma \\ 0 & I \end{bmatrix}\begin{bmatrix} P^* & 0 \\ 0 & Q^* \end{bmatrix}\right\|_2 = \left\|\begin{bmatrix} I & \Sigma \\ 0 & I \end{bmatrix}\right\|_2.$$

But $\left[\begin{smallmatrix} I & \Sigma \\ 0 & I \end{smallmatrix}\right]$ can be permuted into the form $\operatorname{diag}(D_i)$, where $D_i = \left[\begin{smallmatrix} 1 & \sigma_i \\ 0 & 1 \end{smallmatrix}\right]$. It is easy to find the singular values of D_i, and the maximum value is attained for $\sigma_1 = \|F\|_2$.

6.11 (a)

$$\|Ax\|_\beta = \Big\|\sum_j x_j A(:,j)\Big\|_\beta \le \max_j \|A(:,j)\|_\beta \sum_j |x_j|,$$

with equality for $x = e_k$, where the maximum is attained for $j = k$.

(b)

$$\|Ax\|_\infty = \max_i |A(i,:)x| \le \max_i\left(\|A(i,:)\|_\alpha^D \|x\|_\alpha\right) = \|x\|_\alpha \max_i \|A(i,:)\|_\alpha^D.$$

Equality is attained for an x that gives equality in the Holder inequality involving the kth row of A, where the maximum is attained for $i = k$. Finally, from either formula, $\|A\|_{1,\infty} = \max_{i,j} |a_{ij}|$.

6.12. Using the Cholesky factorization $A = R^*R$,

$$\begin{aligned} \|A\|_{\infty,1} &= \max_{x\ne 0} \frac{\|Ax\|_1}{\|x\|_\infty} = \max\{\, y^*Ax : \|x\|_\infty = 1,\ \|y\|_\infty = 1\,\} \\ &= \max\{\, (Ry)^*Rx : \|x\|_\infty = 1,\ \|y\|_\infty = 1\,\} \\ &= \max\{\, (Rx)^*Rx : \|x\|_\infty = 1\,\} \\ &= \max\{\, x^*Ax : \|x\|_\infty = 1\,\}. \end{aligned}$$

6.13. H^T is also a Hadamard matrix, so by the duality result (6.21) it suffices to show that $\|H\|_p = n^{1/p}$ for $1 \le p \le 2$. Since $|h_{ij}| \equiv 1$, (6.12) gives $\|H\|_p \ge n^{1/p}$. Since $\|H\|_1 = n$ and $\|H\|_2 = n^{1/2}$, (6.20) gives $\|H\|_p \le n^{1/p}$, and so $\|H\|_p = n^{1/p}$ for $1 \le p \le 2$, as required.

6.14. We prove the lower bound in (6.22) and the upper bound in (6.23); the other bounds follow on using $\|A^T\|_p = \|A\|_q$. First, note that $\|A\|_p \ge \|Ae_j\|_p = \|A(:,j)\|_p$, which gives the lower bound in (6.22). Now assume that A has at most μ nonzeros per column. Define

$$D_i = \operatorname{diag}(s_{i1}, \dots, s_{in}), \qquad s_{ij} = \begin{cases} 1, & a_{ij} \ne 0, \\ 0, & a_{ij} = 0, \end{cases}$$

and note that

$$\sum_i \|D_i x\|_p^p = \sum_{i,j} s_{ij}|x_j|^p = \sum_j |x_j|^p \sum_i s_{ij} \le \mu\|x\|_p^p.$$

We have

$$\begin{aligned}\|Ax\|_p^p &= \sum_i |A(i,:)x|^p = \sum_i |A(i,:)D_i x|^p \\ &\le \sum_i \|A(i,:)\|_q^p \|D_i x\|_p^p \\ &\le \max_i \|A(i,:)\|_q^p \sum_i \|D_i x\|_p^p \\ &\le \max_i \|A(i,:)\|_q^p \, \mu\|x\|_p^p,\end{aligned}$$

which gives the upper bound in (6.23).

6.15. The lower bound follows from $\|Ax\|_p/\|x\|_p \le \|\,|A||x|\,\|_p/\|\,|x|\,\|_p$. From (6.12) we have

$$\|\,|A|\,\|_p \le n^{1-1/p} \max_j \|\,|A|e_j\|_p = n^{1-1/p} \max_j \|Ae_j\|_p \le n^{1-1/p}\|A\|_p.$$

By (6.21), we also have $\|\,|A|\,\|_p = \|\,|A^T|\,\|_q \le n^{1-1/q}\|A^T\|_q = n^{1/p}\|A\|_p$ and the result follows.

6.16. The function v is not a vector norm because $\nu(\alpha x) \neq |\alpha|\nu(x)$ does not hold for all $\alpha \in \mathbb{C}$, $x \in \mathbb{C}^n$. However, $\nu(\alpha x) \le \nu(\alpha)\nu(x)$, and the other two norm conditions hold, so it makes sense to define the "subordinate norm". We have

$$\begin{aligned}\nu(Ax) &= \nu\Big(\sum_j x_j A(:,j)\Big) \le \sum_j \nu(x_j A(:,j)) \\ &\le \sum_j \nu(x_j)\nu(A(:,j)) \le \max_j \nu(A(:,j)) \sum_j \nu(x_j) \\ &= \max_j \nu(A(:,j))\nu(x).\end{aligned}$$

There is equality for x_j an appropriate unit vector e_j. Hence $v(A) = max_j\ v(A(:,j))$.

7.1. It is straightforward to obtain from $A(y - x) = \Delta b - \Delta Ax + \Delta A(x - y)$ the inequality $(I-\epsilon|A^{-1}|E)|x-y| \le \epsilon|A^{-1}|(f+E|x|)$. In general, if $B \ge 0$ and $r(B) < 1$ then $I - B$ is nonsingular. Since $\rho(\epsilon|A^{-1}|E) \le \epsilon\|\,|A^{-1}|E\,\| < 1$, we can premultiply by $(I - \epsilon|A^{-1}|E)^{-1} \ge 0$ to obtain the bound for $|x - y|$. For the last part,

$$\frac{|x_i - y_i|}{|x_i|} \le \frac{\epsilon\big(|A^{-1}|(f + E|x|)\big)_i}{|x_i|} + O(\epsilon^2).$$

7.2. Take norms in $r = A(x-y)$ and $x-y = A^{-1}r$. The result says that the normwise relative error is at least as large as the normwise relative residual and possibly $\kappa(A)$ times as large. Since the upper bound is attainable, the relative residual is not a good predictor of the relative error unless A is very well conditioned.

7.3. Let D_R equilibrate the rows of A, so that $B = D_R A$ satisfies $|B|e = e$. Then

$$\mathrm{cond}(B) = \||B^{-1}||B|e\|_\infty = \||B^{-1}|e\|_\infty = \|B^{-1}\|_\infty = \kappa_\infty(B).$$

Hence $\kappa_\infty(D_R A) = \mathrm{cond}(D_R A) = \mathrm{cond}(A)$, which implies $\kappa_\infty(A) \le \kappa_\infty(D_R)\,\mathrm{cond}(A)$. The inequality $\mathrm{cond}(A) \le \kappa_\infty(A)$ is trivial, and the deduction of (7.12) is immediate.

7.4. The first inequality is trivial. For the second, since $h_{ii} \equiv 1$ and $|h_{ij}| \le 1$ we have $|H| \ge 1$ and $\|H\|_\infty \le n$. Hence

$$\|\,|H^{-1}||H|\,\|_\infty \ge \|\,|H^{-1}|\,\|_\infty = \|H^{-1}\|_\infty \ge \|H^{-1}\|_\infty \frac{\|H\|_\infty}{n} = \frac{\kappa_\infty(H)}{n}.$$

7.5. We have $\Delta x = A^{-1}\Delta b$, which yields $\|\Delta x\|_2 \le \sigma_n^{-1}\|\Delta b\|_2$. Now

$$\begin{aligned}\|x\|_2^2 = \|V\Sigma^{-1}U^T b\|_2 &= \sum_{i=1}^{n} \frac{(u_i^T b)^2}{\sigma_i^2} \qquad (u_i = U(:,i))\\ &\ge \sum_{i=n+1-k}^{n} \frac{(u_i^T b)^2}{\sigma_i^2}\\ &\ge \frac{1}{\sigma_{n+1-k}^2} \sum_{i=n+1-k}^{n} (u_i^T b)^2 = \frac{\|P_k b\|_2^2}{\sigma_{n+1-k}^2},\end{aligned}$$

which yields the result.

If we take $k = n$ we obtain the bound that would be obtained by applying standard perturbation theory (Theorem 7.2). The gist of this result is that the full $\kappa_2(A)$ magnification of the perturbation will not be felt if b contains a significant component in the subspace $\mathrm{span}(U_k)$ for a k such that $\sigma_{n+1-k} \approx \sigma_n$. This latter condition says that x must be a large-normed solution: $\|x\|_2 \approx \|A^{-1}\|_2\|b\|_2$.

7.6. (a) Use $\|\,|A^{-1}|E|x|\,\|_\infty = \|x\|_1\|\,|A^{-1}||A|e\,\|_\infty = \|x\|_1\|\,|A^{-1}||A|\,\|_\infty$. For the upper bound, use $f \le |A|\,|x|$.

(b) Use $\|\,|A^{-1}|E|x|\,\|_\infty = \|\,|A^{-1}|ee^T|A||x|\,\|_\infty = \|A^{-1}\|_\infty\|\,|A||x|\,\|_1$.

7.7. We will prove the result for w; the proof for h is entirely analogous. The lower bound is trivial. Let $\epsilon = \omega_{|A|,|b|}(y)$ and $r = b - Ay$. Then $|r| \le \epsilon(|A||y| + |b|)$ and $(A + \Delta A)y = b + \Delta b$ with $|\Delta A| \le \epsilon|A|$ and $|\Delta b| \le \epsilon|b|$. Hence $|b| = |(A + \Delta A)y - \Delta b| \le (1+\epsilon)|A||y| + \epsilon|b|$, yielding $|b| \le (1-\epsilon)^{-1}(1+\epsilon)|A||y|$. Thus

$$|r| \le \epsilon\left(1 + \frac{1+\epsilon}{1-\epsilon}\right)|A||y| = \frac{2\epsilon}{1-\epsilon}|A||y|,$$

which implies the result, by Theorem 7.3.

7.8. We have $\Delta x = A^{-1} (Db - \Delta Ax) + O(\epsilon^2)$. Therefore $c^T\Delta x = c^T A^{-1} (\Delta b - \Delta Ax) + O(\epsilon^2)$, and so

$$\frac{|c^T\Delta x|}{\epsilon|c^T x|} \le \frac{|c^T A^{-1}|(f + E|x|)}{|c^T x|} + O(\epsilon),$$

this inequality being sharp. Hence

$$\chi_{E,f}(A,x) = \frac{|c^T A^{-1}|(f + E|x|)}{|c^T x|}.$$

The lower bound for $\chi_{|A|,|b|}(A,x)$ follows from the inequalities $|c^T x| = |c^T A^{-1} \cdot Ax| \le |c^T A^{-1}|\ |A|\ |x|$ and $|c^T x| = |c^T A^{-1} b| < |c^T A^{-1}||b|$. A slight modification to the derivation of $\chi_{E,f}(A, x)$ yields

$$\psi_{E,f}(A,x) = \|c^T A^{-1}\|_2 \frac{\|f\|_2 + \|E\|_2\|x\|_2}{|c^T x|}.$$

7.9. (a) For any $D_1, D_2 \in \mathcal{D}_n$ we have

$$\rho(BC) = \rho(D_1 BC D_1^{-1}) \le \|D_1 BC D_1^{-1}\|_\infty \le \|D_1 B D_2\|_\infty \|D_2^{-1} C D_1^{-1}\|_\infty. \tag{A.4}$$

The rest of the proof shows that equality can be attained in this inequality.

Let $x_1 > 0$ be a right Perron vector of BC, so that $BCx_1 = \pi x_1$, where $\pi = r(BC) > 0$. Define

$$x_2 = Cx_1, \tag{A.5}$$

so that $x_2 > 0$ and

$$Bx_2 = \pi x_1. \tag{A.6}$$

(We note, incidentally, that x_2 is a right Perron vector of CB: $CBx_2 = \pi x_2$.)

Now define

$$D_1 = \mathrm{diag}(x_1)^{-1}, \qquad D_2 = \mathrm{diag}(x_2). \tag{A.7}$$

Then, with $e = [1, 1, \dots, 1]^T$, (A.6) can be written $BD_2 e = \pi D_1^{-1} e$, or $D_1 B D_2 e = \pi e$. Since $D_1 B D_2 > 0$, this gives $\|D_1 B D_2\|_\infty = \pi$. similary, (A.5) can be written $D_2 e = C D_1^{-1} e$, or $D_2^{-1} C D_1^{-1} e = e$, which gives $\|D_2^{-1} C D_1^{-1}\|_\infty = 1$. Hence for D_1 and D_2 defined by (A.7) we have $\|D_1 B D_2\|_\infty \|D_2^{-1} C D_1^{-1}\|_\infty = \pi = \rho(BC)$, as required.

Note that for the optimal D_1 and D_2, $D_1 B D_2$ and $D_2^{-1} C D_1^{-1}$ both have the property that all their rows sums are equal.

(b) Take $B = |A|$ and $C = |A^{-1}|$, and note that $\kappa_\infty(D_1 A D_2) = \|D_1|A|D_2\|_\infty \times \|D_2^{-1}|A^{-1}|D_1^{-1}\|_\infty$. Now apply (a).

(c) We can choose $F_1 > 0$ and $F_2 > 0$ so that $|A| + tF_1 > 0$ and $|A^{-1}| + tF_2 > 0$ for all i! >0. Hence, using (a),

$$\begin{aligned}\inf_{D_1,D_2\in\mathcal{D}_n} \kappa_\infty(D_1 A D_2) &\le \inf_{D_1,D_2\in\mathcal{D}_n} \|D_1(|A| + tF_1)D_2\|_\infty \|D_2^{-1}(|A^{-1}| + tF_2)D_1^{-1}\|_\infty \\ &= \rho\big((|A| + tF_1)(|A^{-1}| + tF_2)\big).\end{aligned}$$

Taking the limit as $t \to 0$ yields the result, using continuity of eigenvalues.

(d) A nonnegative irreducible matrix has positive Perron vector, from standard Perron-Frobenius theory (see, e.g., Horn and Johnson [580, 1985, Chap. 8]). Therefore the result follows by noting that in the proof of (a) all we need is for D_1 and D_2 in (A.7) to be defined and nonsingular, that is, for x_1 and x_2 to be positive vectors. This is the case if BC and CB are irreducible (since x_2 is a Perron vector of CB)

(e) Using $\kappa_1(A) = \kappa_\infty(A^T)$, $\rho(A) = \rho(A^T)$, and $\rho(AB) = \rho(BA)$, it is easy to show that the results of (a)–(d) remain true with the co-norm replaced by the 1-norm. From $\|A\|_2 \le \left(\|A\|_1\|A\|_\infty\right)^{1/2}$ it then follows that $\inf_{D_1,D_2\in\mathcal{D}_n} \kappa_2(D_1AD_2) \le \rho(|A|\,|A^{-1}|)$. In fact, the result in (a) holds for any p-norm, though the optimal D_1 and D_2 depend on the norm; see Bauer [79, 1963, Lem. 1(i)].

7.10. That $\mu'_E(A)$ cannot exceed the claimed expression follows by taking absolute values in the expression $\Delta X = -(A^{-1}\,\Delta AA^{-1} + A^{-1}\,\Delta A\Delta X)$. To show it can equal it we need to show that if the maximum is attained for $(i, j) = (r,s)$ then

$$|\Delta x_{rs}| = \epsilon\left(|A^{-1}|E|A^{-1}|\right)_{rs}$$

can be attained, to first order in ϵ. Equality is attained for $\Delta A = D_1ED_2$, where $D_1 = \mathrm{diag}(\mathrm{sign}(A^{-1})_{ri})$, $D_2 = \mathrm{diag}(\mathrm{sign}(A^{-1})_{is})$.

7.11. (a) We need to find a symmetric H satisfying $Hy = b - Ay =: r$. Consider the QR factorization

$$Q^T\,[y \quad r] = \begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}$$

The constraint can be rewritten $Q^THQ\cdot Q^Ty = Q^Tr$, and it is satisfied by $Q^THQ := \mathrm{diag}(\widetilde{H}, 0_{n-2})$ if we can find $\widetilde{H} \in \mathbb{R}^{2\times2}$ such that

$$\widetilde{H}t = u, \quad \widetilde{H} = \widetilde{H}^T, \text{ where } t = \begin{bmatrix} t_{11} \\ 0 \end{bmatrix}, \quad u = \begin{bmatrix} t_{12} \\ t_{22} \end{bmatrix}.$$

We can take $\widetilde{H} := (\|u\|_2/\|t\|_2)Q$, where $Q \in \mathbb{R}^{2\times2}$ is either a suitably chosen Householder matrix, or the identity if t is a positive multiple of u. Then

$$\begin{aligned} \|H\|_2 &= \|\widetilde{H}\|_2 = \|u\|_2/\|t\|_2 \\ &= \|r\|_2/\|y\|_2 = \|Gy\|_2/\|y\|_2 \le \|G\|_2, \end{aligned}$$

and $\|H\|_F = \|\widetilde{H}\|_F \le \sqrt{2}\|\widetilde{H}\|_2 \le \sqrt{2}\|G\|_2 \le \sqrt{2}\|G\|_F$.

(b) We can assume, without loss of generality, that

$$|y_1| \le |y_2| \le \cdots \le |y_n|. \tag{A.8}$$

Define the off-diagonal of H by $h_{ij} = h_{ji} = g_{ij}$ for $j > i$, and let $h_{11} = g_{11}$. The ith equation of the constraint $Hy = Gy$ will be satisfied if

$$h_{ii}y_i = g_{ii}y_i + \sum_{j=1}^{i-1}(g_{ij} - g_{ji})y_j. \tag{A.9}$$

If $y_i = 0$ set $h_{ii} = 0$; then (A.9) holds by (A.8). Otherwise, set

$$h_{ii} = g_{ii} + \sum_{j=1}^{i-1} (g_{ij} - g_{ji}) \frac{y_j}{y_i}, \tag{A.10}$$

which yields

$$|h_{ii}| \le \epsilon |a_{ii}| + 2\epsilon \sum_{j=1}^{i-1} |a_{ij}| \le 3\epsilon |a_{ii}|,$$

by the diagonal dominance.

(c) Let $D = \mathrm{diag}(a_{ii}^{1/2})$ and $A =: D\tilde{A}D$. Then $\tilde{a}_{ii} \equiv 1$ and $|\tilde{a}_{ij}| \le 1$ for $i \ne j$. The given condition can be written $(\tilde{A} + \tilde{G})\tilde{y} = \tilde{b}$, $|\tilde{G}| \le \epsilon|\tilde{A}|$, where $\tilde{y} := Dy$ and $\tilde{b} := D^{-1}b$. Defining $\tilde{H}$ as in the proof of (b), we find from (A. 10) that $|\tilde{h}_{ii}| \le \epsilon + 2\epsilon(i-1) \le (2n-1)\epsilon$, so that $|\tilde{H}| \le (2n-1)|\tilde{A}|$. With $H := D\tilde{H}D$, we find that $H = H^T$, $(A + H)y = b$, and $|H| \le (2n-1)\epsilon|A|$.

7.12. By examining the error analysis for an inner product (33.1), it is easy to see that for any $x \in \mathbb{R}^n$,

$$|fl(Ax) - Ax| \le \mathrm{diag}(\gamma_{w_i})|A||x|.$$

Hence $\hat{r} = fl(b - A\hat{x})$ satisfies

$$\hat{r} = r + \Delta r, \qquad |\Delta r| \le \Gamma(|A||\hat{x}| + |b|), \tag{A.11}$$

which is used in place of (7.26) to obtain the desired bound.

7.13. (a) Writing $A^{-1} = (\alpha_{ij})$, we have

$$\Delta x_i = -\sum_{j=1}^{n} \alpha_{ij} \sum_{k=1}^{n} \epsilon_{jk} e_{jk} x_k + \sum_{j=1}^{n} \alpha_{ij} \delta_j f_j.$$

By the independence of the ϵ_{ij} and δ_i.

$$\mathcal{E}(\Delta x_i^2) = \sigma^2 \left(\sum_{j=1}^{n} \sum_{k=1}^{n} \alpha_{ij}^2 e_{jk}^2 x_k^2 + \sum_{j=1}^{n} \alpha_{ij}^2 f_j^2 \right) = \sigma^2 \left([A^{-1}][E][x] + [A^{-1}][f] \right)_i.$$

Hence

$$\mathcal{E}(\|\Delta x\|_2^2) = \sum_{i=1}^{n} \mathcal{E}(\Delta x_i^2) = \sigma^2 \| [A^{-1}][E][x] + [A^{-1}][f] \|_1.$$

(b) A traditional condition number would be based on the maximum of $\|\Delta x\|_2/(\sigma\|x\|_2)$ over all "perturbations to the data of relative size s". The expression condexp(A, b) uses the "expected perturbation" $\sqrt{\mathcal{E}(\|\Delta x\|_2^2)}/(\sigma\|x\|_2)$ rather than the worst case.

(c) For $e_{ij} \equiv \|A\|_2$ and $f_i \equiv \|b\|_2$ we have

$$[A^{-1}][E][x] + [A^{-1}][f] = \left(\|A\|_2^2 \|x\|_2^2 + \|b\|_2^2 \right) [A^{-1}] \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix},$$

so

$$\frac{\left(\||[A^{-1}][E][x] + [A^{-1}][f]\||_1\right)^{1/2}}{\|x\|_2} = \left(\|A\|_2^2 + \frac{\|b\|_2^2}{\|x\|_2^2}\right)^{1/2} \|A^{-1}\|_F.$$

But

$$\kappa_{A,b}(A,x) = \left(\|A\|_2 + \frac{\|b\|_2}{\|x\|_2}\right) \|A^{-1}\|_2.$$

Hence

$$2^{-1/2}\kappa_{A,b}(A,x) \le \mathrm{condexp}(A,x) \le \sqrt{n}\,\kappa_{A,b}(A,x).$$

This inequality shows that when perturbations are measured normwise there is little difference between the average and worst-case condition numbers.

7.14. We have

$$\begin{aligned}\|A \circ A^{-T}\|_2 &= \|D_1AD_2 \circ D_1^{-1}A^{-T}D_2^{-1}\|_2 = \|(D_1AD_2) \circ (D_1AD_2)^{-T}\|_2\\ &\le \|D_1AD_2\|_2\|(D_1AD_2)^{-T}\|_2 = \kappa_2(D_1AD_2).\end{aligned}$$

Note that this result gives a lower bound for the optimal condition number, while Bauer's result in Problem 7.9(c) gives an upper bound. There is equality for diagonal A, trivially. For triangular A there is strict inequality in general since the lower bound is 1!

8.1. "Straightforward.

8.2. Let

$$T(\lambda) = \begin{bmatrix} \lambda^{-1} & 1 & 1 \\ 0 & \lambda^{-1} & \lambda^{-1} \\ 0 & 0 & \lambda^{-2} \end{bmatrix}.$$

Then

$$T(\lambda)^{-1} = \begin{bmatrix} \lambda & -\lambda^2 & 0 \\ 0 & \lambda & -\lambda^2 \\ 0 & 0 & \lambda^2 \end{bmatrix}, \quad M(T(\lambda))^{-1} = \begin{bmatrix} \lambda & \lambda^2 & 2\lambda^3 \\ 0 & \lambda & \lambda^2 \\ 0 & 0 & \lambda^2 \end{bmatrix}.$$

As $\lambda \to \infty$, $\|M(T(\lambda))^{-1}\|/\|T(\lambda)^{-1}\| \sim \lambda$. This 3 x 3 example can be extended to an n x n one by padding with an identity matrix.

8.3. The bound follows from Theorem 8.9, since $\left(M(U)^{-1}\right)_{ij} \le 2^{j-i-1}$ $(j > i)$. Using $\|b\|_\infty \le \|U\|_\infty\|x\|_\infty \le n\|x\|_\infty$ we get a similar result to that given by Theorem 8.7.

8.4. Assume T is upper triangular, and write $T = D - U$, where $D = \mathrm{diag}(t_{ii}) \ge 0$ and $U \ge 0$ is strictly upper triangular. Then, using the fact that $(D^{-1}U)^n = 0$,

$$\begin{aligned}|T^{-1}||T| &= (I - D^{-1}U)^{-1}D^{-1} \cdot (D + U)\\ &= \sum_{i=0}^{n-1}(D^{-1}U)^i \cdot (I + D^{-1}U)\\ &= \sum_{i=0}^{n-1}(D^{-1}U)^i + \sum_{i=1}^{n-1}(D^{-1}U)^i.\end{aligned}$$

Now $0 \le b = Tx = (D - U)x$, so $Dx \ge Ux$, that is, $x \ge D^{-1}Ux$. Hence

$$|T^{-1}||T|x = \sum_{i=0}^{n-1}(D^{-1}U)^i x + \sum_{i=1}^{n-1}(D^{-1}U)^i x \le (n + n - 1)x,$$

which gives the result, since $x = T^{-1}b \ge 0$.

8.5. Use (8.3).

8.7. *(a)* Write

$$\|A^{-1}\|_\infty = \max_{x \ne 0} \frac{\|A^{-1}x\|_\infty}{\|x\|_\infty} =: \frac{\|b\|_\infty}{\|y\|_\infty},$$

where $Ab = y$. Let $|b_k| = \|b\|_\infty$. From

$$a_{kk}b_k = y_k - \sum_{j \ne k} a_{kj}b_j,$$

it follows that

$$|a_{kk}||b_k| \le \|y\|_\infty + \sum_{j \ne k} |a_{kj}|\|b\|_\infty,$$

and hence that $\|b\|_\infty/\|y\|_\infty \le 1/\alpha_k \le 1/(\min_i \alpha_i)$.

(b) Apply the result of (a) to AD and use the inequality $\|A^{-1}\|_\infty \le \|(AD)^{-1}\|_\infty \|D\|_\infty$.

(c) If A is triangular then we can achieve any diagonal dominance factors β_i we like, by appropriate column scaling. In particular, we can take $\beta_i \equiv 1$ in (b), which requires $M(A)d = e$, where $d = De$. Then $\|D\|_\infty = \|d\|_\infty = \|M(A)^{-1}e\|_\infty$, so the bound is $\|A^{-1}\|_\infty \le \|M(A)^{-1}e\|_\infty$, as required.

8.8. (a) Using the formula $\det(I + xy^T) = 1 + y^Tx$ we have $\det(A + \alpha e_i e_j^T) = \det(A(I + \alpha A^{-1}e_i e_j^T)) = \det(A)(1 + \alpha e_j^T A^{-1}$. Hence we take $\alpha_{ij} = -(e_j^T A^{-1} e_i)^{-1}$ if $e_j^T A^{-1} e_i \ne 0$; otherwise there is no α_{ij} that makes $A + \alpha_{ij} e_i e_j^T$ singular.

It follows that the "best" place to perturb A to make it singular (the place that gives the smallest α_{ij}) is in the (s, r) position, where the element of largest absolute value of A^{-1} is in the (r, s) position.

(b) The off-diagonal elements of T_n^{-1} are given by $(T_n^{-1})_{ij} = 2^{j-i-1}$. Hence, using part (a), $T_n + \alpha e_n e_1^T$ is singular, where $\alpha = -2^{2-n}$. In fact, T_n is also made singular by subtracting 2^{1-n} from all the elements in the first column.

8.9. Here is Zha's proof. If $s = 1$ the result is obvious, so assume $s < 1$. Define the n-vectors

$$v^T = \left[0, \quad \dots, \quad 0, \quad \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right]$$

and

$$u^T = \left[0, \quad \dots, \quad 0, \quad \sqrt{\frac{1+c}{2}}, \frac{-s}{\sqrt{2(1+c)}}\right]$$

and let $\sigma = s^{n-2}\sqrt{1+c}$. It is easy to check that

$$U_n(\theta)v = \sigma u, \quad U_n(\theta)^T u = \sigma v,$$

which shows that s is a singular value of $U_n(\theta)$. With σ_i denoting the ith largest singular value,

$$\sigma_n(U_n(\theta)) \le s^{n-1} < \sigma = s^{n-2}\sqrt{1+c}.$$

Now we prove by induction that $\sigma_{n-1}(U_n(\theta)) = \sigma$. For $n = 2$ it is easy to check by direct computation that $\sigma_1(U_2(\theta)) = \sqrt{1+c}$. Using the interlacing property of the singular values [470, 1989, §8.3.1] and the inductive assumption, we have

$$\sigma_{n-2}(U_n(\theta)) \ge \sigma_{n-2}(U_{n-1}(\theta)) = s^{n-3}\sqrt{1+c} > \sigma.$$

Therefore $\sigma_{n-1}(U_n(\theta)) = \sigma$.

8.10. For a solver of this form, it is not difficult to see that

$$|fl(f_i(T,b)) - f_i(T,b)| \le c_n u \overline{f}_i(M(T),|b|) + O(u^2),$$

where $\overline{f}_i$ denotes f_i with all its coefficients replaced by their absolute values, and where $\overline{f}_i(M(T), |b|)$ is a rational expression consisting entirely of nonnegative terms. This is the required bound expressed in a different notation. An example (admittedly, a contrived one) of a solver that does not satisfy (8.20) is, for a 2 x 2 lower triangular system $Lx = b$,

$$x_1 = b_1/l_{11}, \quad x_2 = \big((b_2 - b_1)(l_{11} + l_{21}) + b_1 l_{11} - b_2 l_{21}\big)/(l_{11}l_{22}).$$

9.1. The proof is by induction. Assume the result is true for matrices of order $n - 1$, and suppose

$$\begin{bmatrix} \alpha & a^T \\ b & C \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l & M \end{bmatrix} \begin{bmatrix} \alpha & a^T \\ 0 & V \end{bmatrix} \in \mathbb{R}^{n\times n}$$

is a unique LU factorization. Then $MV = C - la^T$ has a unique LU factorization, and so by the inductive hypothesis has nonsingular leading principal submatrices of order 1 to $n - 2$. Thus $v_{ii} \ne 0$, $i = 1{:}n - 2$. If $\alpha = 0$ then $b = 0$ and l is arbitrary subject to $C - la^T$ having an LU factorization. But $C - (l + \Delta l)a^T$ has nonsingular leading principal submatrices of order 1 to $n - 2$ for sufficiently small Δl (irrespective of a), so has an LU factorization for sufficiently small Δl. Thus if $\alpha = 0$ we have a contradiction to the uniqueness of the factorization. Hence $\alpha \ne 0$, which completes the proof.

9.2. $A(s)$ fails to have an LU factorization without pivoting only if one of its leading principal submatrices is singular, that is, if $\sigma \in \lambda\big(A(\sigma)(1{:}k,1{:}k)\big)$, for $k \in \{1,2,\dots,n-1\}$. There are thus $1 + 2 + \cdots + (n - 1) = ½(n - 1)n$ “danger” values of u, which may not all be distinct.

9.3. If $0 \notin F(A)$ then all the principal submatrices of A must be nonsingular, so the result follows by Theorem 9.1. Note that A may have a unique LU factorization even when 0 is in the field of values, as shown by the matrices

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \quad (z = e_2) \quad \text{and} \quad A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \quad (z = e_2),$$

so the implication is only one way. Note also that $0 \notin F(A)$ iff $e^{i\theta} A$ has positive definite Hermitian part for some real q (Horn and Johnson [581, 1991, Thin. 1.3.5]).

9.4. The changes are minor. Denoting by $\widehat{P}$ and $\widehat{Q}$ the computed permutations, the result of Theorem 9.3 becomes

$$\widehat{L}\widehat{U} = \widehat{P}A\widehat{Q} + \Delta A, \qquad |\Delta A| \le \gamma_n |\widehat{L}||\widehat{U}|,$$

and that of Theorem 9.4 becomes

$$(A + \Delta A)\widehat{x} = b, \qquad |\Delta A| \le 2\gamma_n \widehat{P}^T |\widehat{L}||\widehat{U}|\widehat{Q}^T.$$

9.5

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & -1 \end{bmatrix} = LU; \qquad |L||U| = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}.$$

9.6. Since A is nonsingular and every submatrix has a nonnegative determinant, the determinantal inequality shows that $\det(A(1\colon p, 1\colon p)) > 0$ for $p = 1\colon n-1$, which guarantees the existence of an LU factorization (Theorem 9.1). That the elements of L and U are nonnegative follows from (9.2).

GE computes $a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik}a_{kj}^{(k)} = a_{ij}^{(k)} - l_{ik}u_{kj} \le a_{ij}^{(k)}$ since $l_{ik}, u_{kj} \ge 0$. Thus $a_{ij} = a_{ij}^{(1)} \ge a_{ij}^{(2)} \ge \cdots \ge a_{ij}^{(r)}$, $r = \min(i,j)$. For $i > j$, $a_{ij}^{(r)} \ge a_{ij}^{(r+1)} = \cdots = a_{ij}^{(n)} = 0$; for $j \ge i$, $a_{ij}^{(r)} = \cdots = a_{ij}^{(n)} = u_{ij} \ge 0$. Thus $0 \le a_{ij}^{(k)} \le a_{ij}$ for all i, j, k and hence $\rho_n \le 1$. But $\rho_n \ge 1$, so $\rho_n = 1$.

9.7. The given fact implies that JAJ is totally nonnegative. Hence it has an LU factorization $JAJ = LU$ with $L \ge 0$ and $U \ge 0$. This means that $A = (JLJ)(JUJ) \equiv \widetilde{L}\widetilde{U}$ is an LU factorization, and $|\widetilde{L}||\widetilde{U}| = LU = JAJ = |A|$.

9.8. We start with Theorem 9.4, and so need to bound $|\widehat{L}||\widehat{U}|$. Now $a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}u_{kj}$, so $|l_{ik}||u_{kj}| \le 2\max_k |a_{ij}^{(k)}|$. Hence

$$\bigl(|L||U|\bigr)_{ij} = \sum_{k=1}^{\min(i,j)} |l_{ik}||u_{kj}| \le 2n \max_k |a_{ij}^{(k)}|,$$

implying

$$\|\,|L||U|\,\|_\infty \le 2n^2 \max_{i,j,k} |a_{ij}^{(k)}| \le 2n^2 \rho_n \|A\|_\infty.$$

Hence (with the same caveat as mentioned after Theorem 9.5)

$$(A + \Delta A)\widehat{x} = b, \qquad \|\Delta A\|_\infty \le 4n^2 \gamma_n \rho_n \|A\|_\infty.$$

9.9. By inspecting the equations in Algorithm 9.2 we see that the computed LU factors satisfy

$$A - \widehat{L}\widehat{U} = \epsilon\, \widehat{l}_{ij}\widehat{u}_{jj} e_i e_j^T.$$

Since $\widehat{L}\widehat{U}\widehat{x} = b$, we have, writing $\alpha = \epsilon\, \widehat{l}_{ij}\widehat{u}_{jj}$ and using the Sherman–Morrison formula (Problem 24.2),

$$\widehat{x} = (A - \alpha e_i e_j^T)^{-1} b = \left(A^{-1} + \frac{\alpha A^{-1} e_i e_j^T A^{-1}}{1 - \alpha e_j^T A^{-1} e_i} \right) b = x + \frac{\alpha A^{-1} e_i x_j}{1 - \alpha e_j^T A^{-1} e_i}.$$

So

$$x - \widehat{x} = -\frac{\epsilon \widehat{l}_{ij}\widehat{u}_{jj}x_j}{1 - \epsilon \widehat{l}_{ij}\widehat{u}_{jj}A^{-1}(j,i)}A^{-1}(:,i).$$

The error $x - \widehat{x}$ is a rational function of ϵ and is zero if $x_j = 0$, but it will typically be of order $\epsilon\kappa_\infty(A)\|x\|_\infty$.

9.10. $\alpha(B) = \alpha(A)$, and $B^{-1} = \frac{1}{2}\left[\begin{smallmatrix} A^{-1} & A^{-1} \\ A^{-1} & -A^{-1} \end{smallmatrix}\right]$, so $\beta(B) = \frac{1}{2}\beta(A)$. Hence $\theta(B) = (\alpha(B)\beta(B))^{-1} = 2\theta(A)$. Taking $A = S_n$ $g(2n) \ge \rho^c_{2n}(B) \ge \theta(B) = 2\theta(S_n) = n+1$.

9.11. First, the size or accuracy of the pivots is not the fundamental issue. The error analysis shows that instability corresponds to large elements in the intermediate matrices or in L or U. Second, in $PAQ = LU$, u_{nn}^{-1} is an element of A^{-1} *(see* (9.10)), so it is not necessarily true that the pivoting strategy can force u_{nn} to be small. Third, from a practical point of view it is good to obtain small final pivots because they reveal near singularity of the matrix.

9.12. Because the rows of U are the pivot rows, μ_j is the maximum number of times any element in the jth column of A was modified during the reduction. Since the multipliers are bounded by T^{-1}, the bound for $\max_i$ $|a_{ij}^{(k)}|$ follows easily. Thus

$$p_n \le (1 + \mathrm{T}^{-1})^{\max_j \mu_j}.$$

10.1. Observe that if $z = \alpha e_i - e_j$, where e_i is the ith unit vector, then for any α we have, using the symmetry of A,

$$0 < x^TAx = \alpha^2 a_{ii} - 2\alpha a_{ij} + a_{jj}.$$

Since the right-hand side is a quadratic in α, the discriminant must be negative, that is, $4a_{ij}^2 - 4a_{ii}a_{jj} < 0$, which yields the desired result. This result can also be proved using the Cholesky decomposition and the Cauchy–Schwarz inequality: $A = R^TR \Rightarrow a_{ij} = r_i^Tr_j \le \|r_i\|_2\|r_j\|_2 = (a_{ii}a_{jj})^{1/2}$ (there is strict inequality for $i \ne j$, since $r_i \ne \alpha r_j$). The inequality implies that $|a_{ij}| \le \max(a_{ii}, a_{jj})$, which shows that $\max_{i,j}|a_{ij}| = \max_i a_{ii}$, that is, the largest element of A lies on the diagonal.

10.2. Compute the Cholesky factorization $A = R^TR$, solve $R^Ty = x$, then compute y^Ty. In addition to being more efficient than forming A^{-1} explicitly, this approach guarantees a nonnegative result.

10.3. Let $s = c - \sum_{i=1}^{k-1} a_ib_i$. By Lemma 8.4,

$$\widehat{s}(1 + \theta_{k-1}^{(0)}) = c - \sum_{i=1}^{k-1} a_ib_i(1 + \theta_{k-1}^{(i)}).$$

Then $\widehat{y} = fl(\sqrt{\widehat{s}}) = \sqrt{\widehat{s}}(1+\delta)$, $|\delta| \le u$. Hence

$$\widehat{y}^2(1 + \theta_{k+1}) = \frac{\widehat{y}^2}{(1+\delta)^2}(1 + \theta_{k-1}^{(0)}) = c - \sum_{i=1}^{k-1} a_ib_i(1 + \theta_{k-1}^{(i)}),$$

as required.

10.4. $A = \begin{bmatrix} \alpha & a^T \\ a & C \end{bmatrix} \Rightarrow A^{(2)} = \begin{bmatrix} \alpha & a^T \\ 0 & B \end{bmatrix}$, where $B = C - aa^T/\alpha$. Let $y = [1, \theta z]^T$. Then $0 <$ $Ay = \theta^2 z^T C z + 2\theta z^T a + \alpha$ for all θ and so the discriminant $4(z^T a)^2 - 4\alpha z^T C z < 0$ if $z \neq 0$, that is, $z^T B z = z^T C z - \frac{1}{\alpha}(z^T a)^2 > 0$ (since $\alpha = a_{11} > 0$). This shows that B is positive definite.

By induction, the elimination succeeds (i.e., all pivots are nonzero) and all the reduced submatrices are positive definite. Hence $a_{rr}^{(r)} > 0$ for all r. Therefore

$$a_{kk}^{(r+1)} = a_{kk}^{(r)} - \frac{a_{kr}^{(r)}}{a_{rr}^{(r)}} a_{rk}^{(r)} = a_{kk}^{(r)} - \frac{a_{rk}^{(r)2}}{a_{rr}^{(r)}} \le a_{kk}^{(r)}.$$

Thus $a_{kk} = a_{kk}^{(1)} \ge \cdots \ge a_{kk}^{(k)} > 0$. Since the largest element of a positive definite matrix lies on the diagonal (Problem 10.1), for any i, j, k there exists r such that

$$|a_{ij}^{(k)}| \le a_{rr}^{(k)} \le \cdots \le a_{rr}^{(1)} = a_{rr} \le \max_{i,j} |a_{ij}|,$$

which shows that the growth factor $p_n = 1$ (since $p_n \ge 1$).

10.5. From (10.8),

$$|\widehat{R}^T||\widehat{R}| \le (1 - \gamma_{n+1})^{-1}(a_{ii}^{1/2} a_{jj}^{1/2}) \le (1 - \gamma_{n+1})^{-1}(\max_i a_{ii}),$$

so

$$\| \, |\widehat{R}^T||\widehat{R}| \, \|_M \le (1 - \gamma_{n+1})^{-1} \|A\|_M,$$

as required.

10.6. $W = A_{11}^{-1} A_{12} = R_{11}^{-1} R_{12}$, so $RZ = 0$ and hence $AZ = 0$. Z is of dimension $n \times (n - r)$ and of full rank, so it spans null(A).

10.7. The inequalities (10. 13) follow from the easily verified fact that, for $j \ge k$,

$$a_{jj}^{(k)} = \sum_{i=k}^{\min(j,r)} r_{ij}^2.$$

10.8. Examples of indefinite matrices with nonnegative leading principal minors are

$$\begin{bmatrix} 0 & 0 \\ 0 & -1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

A necessary and sufficient condition for definiteness is that all the *principal minors* of A (of which there are 2^{n-1}) be nonnegative (not just the leading principal minors); see, e.g., Horn and Johnson [580, 1985, p. 405] or Mirsky [763, 1961, p. 405] (same page number in both books!).

10.9. For the matrix $Z = [-(R_{11}^{-1} R_{12})^T, I]^T \in \mathbb{R}^{n \times (n-k)}$ we have, from (10.15), $Z^T A Z = S_k(A)$ and so we can take $p = Ze_l$, the first column of Z.

10.10. Theorem 10.3 is applicable only if Cholesky succeeds, which can be guaranteed only if the (suitably scaled) matrix A is not too ill conditioned (Theorem 10.7). Therefore the standard analysis is *not* applicable to positive semidefinite matrices that are very close to being singular. Theorem 10.14 provides a bound on the residual after rank(A) stages and, in particular, on the computed Schur complement, which would be zero in exact arithmetic. The condition of Theorem 10.7 ensures that all the computed Schur complements are positive definite, so that even if $\|W\|_2^2$ magnification of errors occurs, it is absorbed by the next part of the Cholesky factor. The proposed appeal to continuity is simply not valid.

10.11. The analysis in §10.4 shows that for a 2 x 2 pivot E, $\det(E) \le (\alpha^2 - 1)\mu_0^2$ for complete pivoting and $\det(E) \le (\alpha^2 - 1)\lambda^2$ for partial pivoting. Now $\alpha^2 - 1 < 0$ and μ_0 and λ are nonzero if a 2 x 2 pivot is needed. Hence $\det(E) < 0$, which means that E has one positive and one negative eigenvalue. Note that if A is positive definite it follows that all pivots are 1 x 1.

If the block diagonal factor has p_+ positive 1 x 1 diagonal blocks, p_- negative 1 x 1 diagonal blocks, p_0 zero 1 x 1 diagonal blocks, and q 2 x 2 diagonal blocks, then the inertia is $(+, -, 0) = (p_+ + q, p_- + q, p_0)$.

Denote a 2 x 2 pivot by

$$E = \begin{bmatrix} a & b \\ b & c \end{bmatrix},$$

and consider partial pivoting. We know $\det(E) = ac - b^2 < 0$ and $|b| \ge |a|$, so the formula $\det(E) = [(a/b)c - b]b$ minimizes the risk of overflow. Similarly, the formula

$$E^{-1} = \frac{1}{b[(a/b)(c/b) - 1]} \begin{bmatrix} c/b & -1 \\ -1 & a/b \end{bmatrix}$$

helps to avoid overflow; this is the formula used in LINPACK'S `xSIDI` and LAPACK'S `xSYTRI`. The same formulae are suitable for complete pivoting because then $|b| \ge \max(|a|, |c|)$.

10.12. The partial pivoting strategy simplifies as follows: if $|a_{11}| \ge \alpha|a_{21}|$ use a 1 x 1 pivot all, if $|a_{22}| \ge \alpha|a_{12}|$ use a 1 x 1 pivot a_{22}, else use a 2 x 2 pivot, that is, do nothing.

10.13. There may be interchanges, because the tests $|a_{11}| \ge \alpha\lambda$ and $\alpha\lambda^2 \le |a_{11}|\sigma$ can both fail, for example for $A = \left[\begin{smallmatrix} 1 & \theta \\ \theta & 2\theta^2 \end{smallmatrix}\right]$ with $\theta > \alpha^{-1}$. But there can be no 2 x 2 pivots, as they would be indefinite (Problem 10.11). Therefore the factorization is $PAP^T = LDL^T$ for a diagonal D with positive diagonal entries.

10.14. That the growth factor bound is unchanged is straightforward to check. No 2x2 pivots are used for a positive definite matrix because, as before (Problem 10.11), any 2x2 pivot is indefinite. To show that no interchanges are required for a positive definite matrix we show that the second test, $\alpha\lambda^2 \le |a_{11}|\sigma$ is always passed. The submatrix $\left[\begin{smallmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{smallmatrix}\right]$ is positive definite, so $a_{11}a_{rr} - a_{r1}^2 > 0$. Hence

$$|a_{11}|\sigma \ge a_{11}a_{rr} > a_{r1}^2 = \lambda^2 \ge \alpha\lambda^2,$$

as required.

10.15. With partial pivoting the diagonal pivoting method produces the factorization, with $P = I$,

$$\begin{bmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 0 & 1 & \\ 1/\epsilon & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1/\epsilon \\ & 1 & 0 \\ & & 1 \end{bmatrix}.$$

As $\epsilon \rightarrow 0$, $\|L\|_\infty \rightarrow \infty$. In contrast, complete pivoting yields

$$PAP^T = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & \epsilon \\ 0 & \epsilon & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ 1 & 1 & \\ & -\epsilon & 1 \end{bmatrix} \begin{bmatrix} 1 & & \\ & -1 & \\ & & 1+\epsilon^2 \end{bmatrix} \begin{bmatrix} 1 & 1 & \\ & 1 & -\epsilon \\ & & 1 \end{bmatrix},$$

and now $\|L\|_\infty$ is bounded independently of ϵ.

10.16. For any nonzero x we have

$$0 < [x_1^T \quad x_2^T] \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

Putting $x_1 = -A_{11}^{-1}A_{12}x_2$ we obtain

$$0 < [x_1^T \quad x_2^T] \begin{bmatrix} 0 \\ -A_{21}A_{11}^{-1}A_{12}x_2 + A_{22}x_2 \end{bmatrix} = x_2^T(A_{22} - A_{21}A_{11}^{-1}A_{12})x_2,$$

which shows that S is positive definite.

10.17. (a) The nonsingularity of A follows from the factorization

$$\begin{bmatrix} H & B^T \\ B & -G \end{bmatrix} = \begin{bmatrix} I & 0 \\ BH^{-1} & I \end{bmatrix} \begin{bmatrix} H & B^T \\ 0 & -(G + BH^{-1}B^T) \end{bmatrix},$$

since $G + BH^{-1}B^T$ is symmetric positive definite.

(b) It suffices to show that the first $n + m - 1$ leading principal submatrices of $\Pi^T A \Pi$ are nonsingular for any permutation matrix Π. But these submatrices are of the form $\overline{\Pi}^T A_p \overline{\Pi}$, where A_p is a principal submatrix of A and $\overline{\Pi}$ is a permutation matrix. Any such A_p is of the form

$$A_p = \begin{bmatrix} H_p & B_p^T \\ B_p & -G_p \end{bmatrix},$$

where H_p and G_p are principal submatrices of H and G, respectively, and so are positive definite. Thus A_p is symmetric quasidefinite and hence nonsingular by (a), as required.

(c)

$$AS = \begin{bmatrix} H & -B^T \\ B & G \end{bmatrix},$$

so $(AS + (AS)^T)/2 = \mathrm{diag}(H, G)$, which is symmetric positive definite.

11.1

$$|A||x| \le \|A\|_\infty \|x\|_\infty e \le \|A\|_\infty \|x\|_\infty \frac{|x|}{\min_i |x_i|} = \|A\|_\infty \frac{\max_i |x_i|}{\min_i |x_i|} |x|.$$

11.2. The inequality (11.4) yields, with $x_0 := 0$, and dropping the subscripts on the g_i and G_i,

$$|x - \widehat{x}_2| \le G|x - \widehat{x}_1| + g \le G^2|x - \widehat{x}_0| + Gg + g = G^2|x| + (G + I)g.$$

Now $G \approx |F| \le 2\gamma_n |A^{-1}||\widehat{L}||\widehat{U}| \lesssim 2\gamma_n |A^{-1}||B|A|$, where $B = |\widehat{L}||\widehat{L}^{-1}|$ can be assumed to have a modest ∞-norm. Now, using Problem 11.1,

$$\begin{aligned} G^2|x| &\lesssim 2n^2u^2|A^{-1}|(B|A||A^{-1}|B)|A||x| \\ &\lesssim 2n^2u^2\| B|A||A^{-1}|B\|_\infty \sigma(A,x)|A^{-1}||A||x| \\ &\le 2n^2\|B\|_\infty^2 \cdot u\,\mathrm{cond}(A^{-1})\sigma(A,x) \cdot u|A^{-1}||A||x| \\ &\lesssim 2n^2\|B\|_\infty^2 \cdot u|A^{-1}||A||x|, \end{aligned}$$

under the conditions of Theorem 11.4. The term G_g can be bounded in a similar way. The required bound for $\|x - \widehat{x}_2\|_\infty / \|x\|_\infty$ follows.

12.1. The equations for the blocks of L and U are $U_{11} = A_{11}$ *and*

$$\left.\begin{aligned} L_{i,i-1} &= A_{i,i-1}U_{i-1,i-1}^{-1} \\ U_{ii} &= A_{ii} - L_{i,i-1}A_{i-1,i} \end{aligned}\right\} \quad i \ge 2.$$

First, consider the case where A is block diagonally dominant by columns. We prove by induction that

$$\|A_{k,k-1}\|\,\|U_{k-1,k-1}^{-1}\| \le 1, \quad k \ge 2.$$

which implies both the required bounds. This inequality clearly holds for $k = 2$; suppose it holds for $k = i$. We have

$$U_{ii} = A_{ii}(I - A_{ii}^{-1}L_{i,i-1}A_{i-1,i}) =: A_{ii}(I - P),$$

where

$$\begin{aligned} \|P\| &\le \|A_{ii}^{-1}\|\big(\|A_{i,i-1}\|\,\|U_{i-1,i-1}^{-1}\|\big)\|A_{i-1,i}\| \\ &\le \|A_{ii}^{-1}\|\,\|A_{i-1,i}\| \\ &\le 1 - \|A_{ii}^{-1}\|\,\|A_{i+1,i}\|, \end{aligned}$$

using the block diagonal dominance for the last inequality. Hence

$$\|U_{ii}^{-1}\| \le \frac{\|A_{ii}^{-1}\|}{1 - \|P\|} \le \frac{1}{\|A_{i+1,i}\|},$$

as required.

The proof for A block diagonally dominant by rows is similar. The inductive hypothesis is that

$$\|U_{k-1,k-1}^{-1}\|\,\|A_{k-1,k}\| \le 1,$$

and with P defined as before, we have

$$\begin{aligned}\|P\| &\le \|A_{ii}^{-1}\|\,\|A_{i,i-1}\|\,\|U_{i-1,i-1}^{-1}\|\,\|A_{i-1,i}\|\\ &\le \|A_{ii}^{-1}\|\,\|A_{i,i-1}\|\\ &\le 1-\|A_{ii}^{-1}\|\,\|A_{i,i+1}\|,\end{aligned}$$

giving $\|U_{ii}^{-1}\| \le \|A_{i,i+1}\|^{-1}$, as required.

For block diagonal dominance by columns in the ∞-norm we have $\|L\|_\infty \le 2$ and $\|U\|_\infty \le 3\|A\|_\infty$, so block LU factorization is stable. If A is block diagonally dominant by rows, stability is assured if $\|A_{i,i-1}\|/\|A_{i-1,i}\|$ is suitably bounded for all a.

12.2. The block 2 x 2 matrix

$$\left[\begin{array}{cc|cc} \epsilon & 2 & 1 & 0\\ 2 & \epsilon & 0 & 1\\ \hline 1 & 0 & \epsilon & 2\\ 0 & 1 & 2 & \epsilon \end{array}\right]$$

is block diagonally dominant by rows and columns in the 1- and ∞-norms for $\epsilon = 1/2$, but is not point diagonally dominant by rows or columns. The block 2 x 2 matrix

$$\left[\begin{array}{cc|cc} 2 & 1 & 1 & 0\\ 1 & 1 & 0 & 0\\ \hline 1 & 0 & 2 & 1\\ 0 & 0 & 1 & 1 \end{array}\right]$$

is point diagonally dominant by rows and columns but not block diagonally dominant by rows or columns in the ∞-norm or l-norm.

12.3. No. A counterexample is the first matrix in the solution to Problem 12.2, with $\epsilon = 1/2$, which is clearly not positive definite because the largest element does not lie on the diagonal.

12.4. Form (12.2) it can be seen that $(A^{-1})21 = -S^{-1}A_{21}A_{11}^{-1}$, where the Schur complement $S = A_{22} - A_{21}A_{11}^{-1}A_{12}$. Hence

$$\|A_{21}A_{11}^{-1}\| \le n\|S\|\,\|(A^{-1})_{21}\| \le n\|S\|\,\|A^{-1}\|.$$

S is the trailing submatrix that would be obtained after $r-1$ steps of GE. It follows immediately that $\|S\| \le p_n\,\|A\|$.

For the last part, note that $\|S^{-1}\| \le \|A^{-1}\|$, because S^{-1} is the (2,2) block of A^{-1}, *as* is easily seen from (12.2).

12.5. The proof is similar to that of Problem 8.7(a). We will show that $\|A_{11}^{-T}A_{21}^T\|_\infty \le 1$. Let $y = A_{11}^{-T}A_{21}^T x$ and let $|y_k| = \|y\|_\infty$. The kth equation of $A_{11}^T y = A_{21}^T x$ gives

$$a_{kk}y_k = \sum_{i=r+1}^{n} a_{ik}x_{i-r} - \sum_{\substack{i=1\\ i\ne k}}^{r} a_{ik}y_i.$$

Hence

$$\|y\|_\infty = |y_k| \le \|x\|_\infty \sum_{i=r+1}^{n} \left|\frac{a_{ik}}{a_{kk}}\right| + \|y\|_\infty \sum_{\substack{i=1 \\ i\ne k}}^{r} \left|\frac{a_{ik}}{a_{kk}}\right|,$$

which yields $\|y\|_\infty \le \|x\|_\infty$ in view of the column diagonal dominance, as required.

12.7. We have the block LU factorization

$$X = \begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & D - CA^{-1}B \end{bmatrix},$$

so that

$$\det(X) = \det(A) \times \det(D - CA^{-1}B).$$

Hence $\det(X) = \det(AD - ACA^{-1}B)$, which equals $\det(AD - CB)$ if C commutes with A.

13.2. The new bounds have norms in place of absolute values and the constants are different.

13.3. Immediate from $AX - I = A(XA - I)A^{-1}$.

13.4. With $0 < \epsilon \ll 1$, let

$$A = \begin{bmatrix} \frac{1}{\epsilon} & 1 \\ \frac{1}{\epsilon^2} - 1 & \frac{1}{\epsilon} \end{bmatrix}, \qquad X = \begin{bmatrix} 1 - \epsilon + \frac{2}{\epsilon} & -2 - \epsilon \\ 2 - \epsilon + \frac{1}{\epsilon} - \frac{1}{\epsilon^2} & -1 - \epsilon + \frac{1}{\epsilon} \end{bmatrix}.$$

Then

$$AX = \begin{bmatrix} O(\epsilon^{-2}) & O(\epsilon^{-1}) \\ O(\epsilon^{-3}) & O(\epsilon^{-2}) \end{bmatrix}$$

while

$$XA = \begin{bmatrix} 1 + \epsilon & -\epsilon \\ \epsilon & 1 - \epsilon \end{bmatrix}.$$

Hence $\|AX - I\|/\|XA - I\| \to \infty$ as $\epsilon \to 0$. Note that in this example *every* element of $AX - I$ is large.

13.5. We have $\widehat{x} = fl(\widehat{X}b) = \widehat{X}b + f_1$, where $|f_1| \le \gamma_n|\widehat{X}||b|$. Hence $A\widehat{x} = A\widehat{X}b + Af_1 = b + r_1$ where $|r_1| \le u|A||\widehat{X}||b| + \gamma_n|A||\widehat{X}||b| \le \gamma_{n+1}|A||\widehat{X}||b|$. Similarly,

$$\widehat{y} = fl(\widehat{Y}b) = \widehat{Y}b + f_2, \qquad |f_2| \le \gamma_n|\widehat{Y}||b|. \tag{A.12}$$

Hence $A\widehat{y} = A\widehat{Y}b + Af_2 = A(\widehat{Y}A)x + Af_2 = b + r_2$, where

$$|r_2| \le u|A||\widehat{Y}||A||x| + \gamma_n|A||\widehat{Y}||b| \le \gamma_{n+1}|A||\widehat{Y}||A||x|.$$

Clearly,

$$|x - \widehat{x}| \le (n+1)u|A^{-1}||A||A^{-1}||b| + O(u^2).$$

From (A.12), $\widehat{y} = \widehat{Y}Ax + f_2$, so

$$|y - \widehat{y}| \le u|\widehat{Y}||A||x| + \gamma_n|\widehat{Y}||b| \le (n+1)u|A^{-1}||A||x| + O(u^2).$$

The first conclusion is that the approximate left inverse yields the smaller residual bound, while the approximate right inverse yields the smaller forward error bound. Therefore which inverse is "better" depends on whether a small backward error or a small forward error is desired. The second conclusion is that neither approximate inverse yields a componentwise backward stable algorithm, despite the favorable assumptions on $\widehat{X}$ and $\widehat{Y}$. Multiplying by an explicit inverse is simply not a good way to solve a linear system.

13.6. Here is a hint: notice that the matrix on the front cover of the *LAPACK Users' Guide* has the form

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \mathrm{diag}(L, A, P, A, C, K).$$

13.7. If the ith row of A contains all 1s then simply sum the elements in the ith row of the equation $AA^{-1} = I$.

13.8. $(A + iB)(P + iQ) = I$ is equivalent to $AP - BQ = I$ and $AQ + BP = 0$, or

$$\begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} P \\ Q \end{bmatrix} = \begin{bmatrix} I \\ 0 \end{bmatrix},$$

so X^{-1} is obtainable from the first n columns of Y^{-1}. The definiteness result follows from

$$\begin{aligned}(x + iy)^*(A + iB)(x + iy) &= x^T(Ax - By) + y^T(Ay + Bx) \\ &\quad + i[x^T(Ay + Bx) - y^T(Ax - By)] \\ &= [x^T \quad y^T] \begin{bmatrix} A & -B \\ B & A \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},\end{aligned}$$

where we have used the fact that $A = AT$ and $B = -B^T$. Doubling the dimension (from X to Y) multiplies the number of flops by a factor of 8, since the flop count for inversion is cubic in the dimension. Yet complex operations should, in theory, cost between about two and eight times as much as real ones (the extremes being for addition and division). The actual relative costs of inverting X and Y depend on the machine and the compiler, so it is not possible to draw any firm conclusions. Note also that Y requires twice the storage of X.

13.10. As in the solution to Problem 8.8, we have

$$\det(A + \alpha e_i e_j^T) = \det(A)(1 + \alpha e_j^T A^{-1} e_i).$$

If $(A^{-1})_{ji} = 0$, this expression is independent of α, and hence det(A) is independent of a_{ij}. (This result is clearly correct for a triangular matrix.) That det(A) can be independent of a_{ij} shows that det(A), or even a scaling of it, is an arbitrarily poor measure of conditioning, for $A + \alpha e_i e_j^T$ approaches a multiple of a rank-1 matrix as $\alpha \to \infty$.

13.11. That Hadamard's inequality can be deduced from QR factorization was noted by Householder [586, 1958, p. 341]. From $A = QR$ we have $q_k^T A = e_k^T R$, *so* that

$$|r_{kk}| = |q_k^T a_k| \le \|a_k\|_2.$$

Hadamard's inequality follows since |det(A)| = |det(R)| = $|r_{11} \ . \ . \ . \ r_{nn}|$. There is equality when $|q_k^T a_k| = \|a_k\|_2$ for k = 1: n, that is, when $a_k = akqk$, k = 1: n, for some scalars αk. In other words, there is equality when R in the QR factorization is diagonal.

13.12. (a) Straightforward. (b) For $A = U(1)$, $\psi(A) = \sqrt{n!}$. For the Pei matrix, $= (\alpha^2 + n - 1)^{n/2} / \big((n + \alpha - 1)(\alpha - 1)^{n-1}\big)$

13.13. (a) The geometric mean of $\sigma_1^2/2, \sigma_1^2/2, \sigma_2^2, \dots, \sigma_{n-1}^2$ is

$$\left(\frac{1}{4}\sigma_1^4\sigma_2^2 \dots \sigma_{n-1}^2\right)^{1/n} = \left(\frac{1}{4}\frac{\sigma_1^2}{\sigma_n^2}\sigma_1^2\sigma_2^2 \dots \sigma_n^2\right)^{1/n} = \left(\frac{1}{2}\kappa_2(A)|\det(A)|\right)^{2/n}.$$

Since the geometric mean does not exceed the arithmetic mean,

$$\left(\frac{1}{2}\kappa_2(A)|\det(A)|\right)^{2/n} \le \frac{1}{n}(\sigma_1^2 + \dots + \sigma_{n-1}^2) < \frac{1}{n}(\sigma_1^2 + \dots + \sigma_n^2) = \frac{1}{n}\|A\|_F^2,$$

which gives the required bound. (b) is trivial.

13.14. The key observation is that for a triangular system $Tx = b$, $T \in \mathbb{R}^{n\times n}$, the computed solution from substitution satisfies

$$(T + \Delta T)\widehat{x} = b + \Delta b, \qquad |\Delta T| \le \gamma_n |T|, \quad \mathrm{diag}(T) = 0, \quad |\Delta b| \le \gamma_n |b|.$$

This result holds for any order of evaluation and is proved (for any particular order of evaluation) using a variation on the proof of Lemma 8.2 in which we do not divide through by the product of 1 + δ_i terms. Using (A. 13) we have

$$\widehat{d} = (1 + \theta_n)\det(T)\big(\eta - (h + \Delta h)^T (T + \Delta T)^{-1}(y + \Delta y)\big),$$

where $|\theta_n| \le \gamma_n$, $|\Delta y| \le \gamma_{n-1}|y|$, and $|\Delta h| \le \gamma_{n-1}|h|$. But, since diag($\Delta T$) = 0, det$(T)$ = det$(T + \Delta T)$, hence

$$\widehat{d} = \det(T + \Delta T)\big(\eta(1 + \theta_n) - (h + \widetilde{\Delta h})^T (T + \Delta T)^{-1}(y + \Delta y)\big),$$

where $|\Delta h| \le \gamma_{2n-1}|h|$. The conclusion is that $\widehat{d} = \det(H + \Delta H)$, where $|\Delta H| \le \gamma_{2n-1}|H|$.

If $\widehat{f} = fl(\det(D^{-1}HD))$ then

$$\widehat{f} = \det(D^{-1}HD + \Delta), \quad |\Delta| \le \gamma_{2n-1}|D^{-1}HD| = \gamma_{2n-1}|D^{-1}||H||D|.$$

Thus

$$\widehat{f} = \det(D^{-1}(H + D\Delta D^{-1})D) = \det(H + D\Delta D^{-1}), \quad |D\Delta D^{-1}| \le \gamma_{2n-1}|H|.$$

A diagonal similarity therefore has no effect on the error bound and so there is no point in scaling H before applying Hyman's method.

13.15. For $A \in \mathbb{R}^{n\times n}$ we have, for any i ε $\{1,2,\ldots,n\}$,

$$\det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} \det(A_{ij}),$$

where A_{ij} denotes the submatrix of A obtained by deleting row i and column j. A definition of condition number that is easy to evaluate is

$$\begin{aligned}
\operatorname{cond}_{\det}(A) &= \max_{i,j} \lim_{\epsilon\to 0} \left| \frac{(\det(A + \epsilon a_{ij} e_i e_j^T) - \det(A))/\det(A)}{\epsilon} \right| \\
&= \max_{i,j} \lim_{\epsilon\to 0} \left| \frac{\epsilon a_{ij} \det(A_{ij})/\det(A)}{\epsilon} \right| \\
&= \max_{i,j} \left| \frac{a_{ij}\det(A_{ij})}{\det(A)} \right| = \max_{i,j} |a_{ij}(A^{-1})_{ji}|.
\end{aligned}$$

14.3 Straightforward, since L is unit lower triangular with $|l_{ij}| \le 1$.

14.6 Let $D = \operatorname{diag}(d_j)$, where $d_1 = 1$ and

$$d_j = \prod_{k=1}^{j-1} \frac{a_{k,k+1}}{a_{k+1,k}}, \qquad j = 2\colon n.$$

Then $T = DA$ is tridiagonal, symmetric and irreducible. By applying Theorem 14.9 and using symmetry, we find that

$$(A^{-1})_{ij} = \begin{cases} x_i y_j d_j, & i \le j, \\ y_i x_j d_j, & i \ge j. \end{cases}$$

There is one degree of freedom in the vectors x and y, which can be expended by setting $x_1 = 1$, say.

15.1. The equation would imply, on taking traces, 0 = trace(I), which is false.

15.2. It is easily checked that the differential equation given in the hint has the solution $Z(t) = e^{At}Ce^{Bt}$. Integrating the differential equation between 0 and ∞ gives, assuming that $\lim_{t\to\infty} Z(t) = 0$,

$$-C = A\left(\int_0^\infty Z(t)\right)dt + \left(\int_0^\infty Z(t)dt\right)B.$$

Hence $-\int_0^\infty Z(t)\,dt = -\int_0^\infty e^{At}Ce^{Bt}\,dt$ satisfies the Sylvester equation. For the Lyapunov equation, the integral $-\int_0^\infty e^{At}(-C)e^{A^Tt}\,dt$ exists under the assumption on A, and the corresponding quadratic form is easily seen to be positive.

15.3. Since

$$\|AX + XA^T\|_F = \|(AX + XA^T)^T\|_F = \|AX^T + X^TA^T\|_F,$$

X^T is a minimizer if X is. If $X + X^T = 0$ then X is a skew-symmetric minimizer. Otherwise, note that the minimization problem is equivalent to finding the right singular vector corresponding to the smallest singular value $\sigma_{\min}$ of $P = I \otimes A + A \otimes I$. Since vec(X) and vec($X^T$) (suitably normalized) are both singular vectors corresponding to $\sigma_{\min}$, so is their sum, vec$(X + X^T) \neq 0$. Thus the symmetric matrix $X + X^T$ is a minimizer.

Byers and Nash [176, 1987] investigate conditions under which a symmetric minimizer exists.

15.4. XLASY2 uses Gaussian elimination with complete pivoting to solve the 2 x 2 or 4 x 4 linear system that arises on converting to the Kronecker product form (15.2). For complete pivoting on a 4 x 4 matrix the growth factor is bounded by 4 (§9.3), versus 8 for partial pivoting. The reasons for using complete pivoting rather than partial pivoting are that, first, the small increase in cost is regarded as worthwhile for a reduction in the error bound by a factor of 2 and, second, complete pivoting reveals the rank better than partial pivoting, enabling better handling of ill-conditioned systems [38, 1993, p. 78].

16.1. Since $\rho(B) < 1$, a standard result (Problem 6.8) guarantees the existence of a consistent norm $\|\cdot\|_p$ for which $\|B\|_p < 1$. The series $\sum_{k=0}^{\infty} \|B^k\|_p \le \sum_{k=0}^{\infty} \|B\|_p^k = (1 - \|B\|_p)^{-1}$ is clearly convergent, and so, by the equivalence of norms, $\sum_{k=0}^{\infty} \|B^k\|$ is convergent for any norm.

Since $(|B^k|)_{ij} \le \|B^k\|_\infty$, the convergence of $\sum_{k=0}^{\infty} \|B^k\|_\infty$ ensures that of $\sum_{k=0}^{\infty} |B^k|$. (The convergence of $\sum_{k=0}^{\infty} |B^k|$ can also be proved directly using the Jordan canonical form.)

16.2. (a) We have

$$\|x_{k+1} - a\| = \|G(x_k) - a + e_k\| \le \beta\|x_k - a\| + \alpha. \tag{A.14}$$

If $\|x_k - a\| \le \alpha/(1-\beta)$ then

$$\|x_{k+1} - a\| \le \beta\frac{\alpha}{1-\beta} + \alpha = \frac{\alpha}{1-\beta}.$$

If $\|x_k - a\| > \alpha/(1-\beta)$, so that $\beta\|x_k - a\| < \|x_k - a\| - \alpha$, then, using (A.14),

$$\|x_{k+1} - a\| < (\|x_k - a\| - \alpha) + \alpha = \|x_k - a\|.$$

(b) We break the proof into two cases. First, suppose that $\|x_m - a\| \le \alpha/(1-\beta)$ for some m. By part (a), the same inequality holds for all $k \ge m$ and we are finished. Otherwise, $\|x_k - a\| > \alpha/(1-\beta)$ for all k. By part (a) the positive sequence $\|x_k - a\|$ is monotone decreasing and therefore it converges to a limit l. Suppose that $l = \alpha/(1-\beta) + \delta$, where $\delta > 0$. Since $\beta < 1$, for some k we must have

$$\|x_k - a\| < \frac{\alpha}{1-\beta} + \frac{\delta}{\beta}.$$

By (A.14),

$$\|x_{k+1} - a\| < \frac{\beta\alpha}{1-\beta} + \delta + \alpha = \frac{\alpha}{1-\beta} + \delta = l,$$

which is a contradiction. Hence $l = \alpha/(1-\beta)$, as required.

(c) The vectors e_k can be regarded as representing the rounding errors on the kth iteration. The bound in (b) tells us that provided the error vectors are bounded by α, we can expect to achieve a relative error no larger than $\alpha/(1-\beta)$. In other words, the result gives us an upper bound on the achievable accuracy. When this result is applied to stationary iteration, β is a norm of the iteration matrix; to make β less than 1 we may have to use a scaled norm of the form $\|A\| := \|XAX^{-1}\|$.

17.1. Let X_n be the $n \times n$ version of the upper triangular matrix

$$\begin{bmatrix} 1 & -\theta & -\theta & -\theta \\ & 1 & -\theta & -\theta \\ & & 1 & -\theta \\ & & & 1 \end{bmatrix} D^{-1}, \qquad \theta > 0,$$

where $D = \mathrm{diag}(\sqrt{1+(j-1)\theta^2})$, so that $\|X_n(:,j)\|_2 = 1$, $j = 1{:}n$. From (8.3), X_n^{-1} is the obvious $n \times n$ analogue of

$$D\begin{bmatrix} 1 & \theta & \theta(\theta+1) & \theta(\theta+1)^2 \\ & 1 & \theta & \theta(\theta+1) \\ & & 1 & \theta \\ & & & 1 \end{bmatrix}$$

Let $A = \mathrm{diag}(0,\dots,0,\lambda)$, with $|\lambda| < 1$. Then

$$\begin{aligned} \|A^k\|_2 &= \|X_nA^kX_n^{-1}\|_2 = |\lambda|^k\|X_ne_ne_n^TX_n^{-1}\|_2 \\ &= |\lambda|^k\|X_n(:,n)\|_2\|X_n^{-1}(n,:)\|_2 = |\lambda|^k\sqrt{1+(n-1)\theta^2}. \end{aligned}$$

But

$$\kappa_2(X_n)\rho(A)^k \ge |\lambda|^k \max_{i,j}|x_{ij}| \max_{i,j}|(X_n^{-1})_{ij}| \ge |\lambda|^k\theta^{n-1}.$$

Therefore $\kappa_2(X_n)\rho(A)^k/\|A^k\|_2 \to \infty$ as $\theta \to \infty$. There is still the question of whether X_n is optimally scaled. We can perturb the zero eigenvalues of A to distinct, sufficiently small numbers so that $\|A^k\|_2$ is essentially unchanged and so that the only freedom in X_n is a diagonal scaling. Since X_n has columns of unit 2-norm, $\min_{F=\mathrm{diag}(f_i)} \kappa_2(X_nF) \ge n^{-1/2}\kappa_2(X_n)$ (Theorem 7.5), so even for the optimally scaled X_n the bound can be arbitrarily poor.

17.2. A simple example is

$$A = \begin{bmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ \alpha & -\alpha & 1 \end{bmatrix},$$

for which $\lambda(A) = \{-1,1,3\}$ and $\lambda(|A|) = \{-1, 2 \pm (4\alpha+1)^{1/2}\}$, so $\rho(|A|)/\rho(A) \to \infty$ as $\alpha \to \infty$.

18.1. For the Householder matrix $I - (2/v^Tv)vv^T$ there is an eigenvalue -1 with eigenvector v, and there are $n - 1$ eigenvalues 1, with eigenvectors any basis for span$(v)^\perp$. A Givens matrix $G(i,j,q)$ has eigenvalues $e^{\pm i\theta}$, together with $n - 2$ eigenvalues 1.

18.2. Straightforward manipulation shows that a bound holds of the form $cnu + O(u^2)$, where c is a constant of order 10.

18.3. We must have $x^*Px = x^*y$, so x^*y must be real. Also, we need $x^*x = y^*y$ and $x \neq y$. Then, with $v = x - y$, $v^*v = 2v^*x$, so $Px = y$.

LAPACK uses modified Householder matrices $P = I - tvv^*$ that are unitary but not Hermitian (T is not real). The benefit of this approach is that such a P can always be chosen so that $Px = \alpha e_1$, with α real (for this equation to hold for a genuine Householder matrix P it would be necessary that x_1 be real). For more details see Lehoucq [698, 1994].

18.4. False. $\det(P) = -1$ for any Householder matrix and $\det(G) = 1$ for any Givens matrix, so $\det(Q) = 1$. Moreover, whereas P is generally a full, symmetric matrix, Q has some zero entries and is generally nonsymmetric.

18.5. The inequalities follow from the fact that, in the notation of (18.1), $\|x_k\|_2 = |r_{kk}|$ and $\|c_k(:,j)\|_2^2 = \sum_{i=k}^{j} r_{ij}^2$, the latter equality being a consequence of the invariance of the 2-norm under orthogonal transformations. The last part follows because QR factorization with column pivoting on A is essentially equivalent to Cholesky factorization with complete pivoting applied to A^TA.

18.6. If $y = |W|x$ and W comprises r disjoint rotations then, in a rather loose notation,

$$\begin{aligned}\|y\|_2^2 &\le \sum_{k=1}^{r} \left\| \left| \begin{bmatrix} c_k & s_k \\ -s_k & c_k \end{bmatrix} \right| \begin{bmatrix} x_{i_k} \\ x_{j_k} \end{bmatrix} \right\|_2^2 + \sum_{k=2r+1}^{m} x_{i_k}^2 \\ &\le 2\sum_{k=1}^{2r} x_{i_k}^2 + \sum_{k=2r+1}^{m} x_{i_k}^2 \le 2\|x\|_2^2.\end{aligned}$$

18.7. Straightforward. This problem shows that the CGS and MGS methods correspond to two different ways of representing the orthogonal projection onto span$\{q_1, \dots, q_j\}$.

18.8. Assume, without loss of generality, that $\|a_1\|_2 \le \|a_2\|_2$. If E is any matrix such that $A + E$ is rank deficient then

$$0 = \sigma_{\min}(A + E) \ge \sigma_{\min}(A) - \|E\|_2 \quad \Rightarrow \quad \sigma_{\min}(A) \le \|E\|_2.$$

We take $E = [e_1, 0]$, where e_l is chosen so that $e_1^T a_1 = 0$ and $a_1 + e_1 = \alpha a_2$, for some α. From Pythagoras's theorem we have that $\|e_1\|_2 = \tan q\, \|a_1\|_2$, and so

$$\sigma_{\min}(A) \le \tan\theta\, \|a_1\|_2 = \tan\theta\, \min(\|a_1\|_2, \|a_2\|_2).$$

Together with the trivial bound $\sigma_{\max}(A) = \|A\|_2 \ge \max(\|a_1\|_2, \|a_2\|_2)$, this yields the result.

18.9. We find that

$$\widehat{Q}_{CGS} = \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}, \qquad \widehat{Q}_{MGS} = \begin{bmatrix} 1 & 0 & 0 \\ \epsilon & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{6}} \\ 0 & 0 & \sqrt{\frac{2}{3}} \end{bmatrix}.$$

For $\widehat{Q}_{MGS}$, $|\widehat{q}_i^T\widehat{q}_j| \le \epsilon/\sqrt{2}$ for $i \ne j$, but for $\widehat{Q}_{CGS}$, $|\widehat{q}_2^T\widehat{q}_3| = 1/2$. It is easy to see that $fl(1+\epsilon^2) = 1$ implies $\epsilon \le \sqrt{u}$ and that $\kappa_2(A) = \epsilon^{-1}\sqrt{3+\epsilon^2}$. Hence for $\widehat{Q}_{MGS}$, $|\widehat{q}_i^T\widehat{q}_j| \lesssim \kappa_2(A)u/\sqrt{6}$, showing, as expected, that the loss of orthogonality for MGS is bounded in terms of $k_2(A)u$

18.10. P is the product $P_n \dots P_1$, where P_k is defined in (18.20). Since $v_k^Tv_{k-1} = e_k^Te_{k-1} + q_k^Tq_{k-1} = 0$, we have

$$P_kP_{k-1} = (I - v_kv_k^T)(I - v_{k-1}v_{k-1}^T) = I - v_kv_k^T - v_{k-1}v_{k-1}^T.$$

Hence

$$P = I - \sum_{k=1}^{n} v_kv_k^T = I - \sum_{k=1}^{n} \begin{bmatrix} e_ke_k^T & -e_kq_k^T \\ -q_ke_k^T & q_kq_k^T \end{bmatrix}.$$

This is of the required form, with $Q = [q_1, \dots, q_n]$ the matrix from the MGS method applied to A.

18.11. With Q defined as in the hint,

$$\begin{aligned} DA = QR - A &= (Q - P_{21})R + DA_2 \\ &= (VW^T - P_{21})R + DA_2 \\ &= V(I - S)W^TR + DAz \\ &= V(I + S)^{-1}C^2W^TR + DA2 \\ &= V(I + S)^{-l}W^T \cdot WCU^T \cdot UCW^T \cdot R + AAz \\ &= V(I+S)^{-1}W^T \cdot P_{11}^TP_{11}R + \Delta A_2 \\ &= V(I+S)^{-1}W^T \cdot P_{11}^T\Delta A_1 + \Delta A_2, \end{aligned}$$

and the result follows.

18.12. To produce the Householder method's P *we* have to explicitly form the product of the individual Householder transformations. As long as this is done in a stable way the computed P is guaranteed to be nearly orthogonal. MGS's Q is formed in an algebraically different way, and the rounding errors in its formation are different from those in the formation of *P;* in other words, Q is not a submatrix of P. Consequently, there is no reason to expect Q to be nearly orthonormal. Further insight can be gained by a detailed look at the structure of the computed *P; see* Björck and Paige [119, 1992, §4].

18.13. It is straightforward to show that $A^TA - I = (A-U)^T(A+U)$. Taking norms gives the lower bound. Since $A + U = U(H+I)$ we have, from the previous relation,

$$(A-U)^TU = (A^TA - I)(H+I)^{-1}.$$

Hence

$$\|A-U\|_2 = \|(A-U)^TU\|_2 \le \|A^TA - I\|_2 \|(H+I)^{-1}\|_2 \le \|A^TA - I\|_2.$$

In fact, the result holds for any unitarily invariant norm (but the $\|A\|_2 + 1$ in the denominator must be retained).

19.1. One approach is to let x be a solution and y an arbitrary vector, and consider $f(\alpha) = \|A(x+\alpha y) - b\|_2^2$. By expanding this expression it can be seen that if the normal equations are not satisfied then α and y can be chosen so that $f(\alpha) < f(0)$. Alternatively, note that for $f(z) = (b-Ax)^T(b-Ax) = x^TA^TAx - 2b^TAx + b^Tb$ we have $\nabla f(x) = 2(A^TAx - A^Tb)$ and $\nabla^2 f(x) = 2A^TA$, so any solution of the normal equations is a local minimum of f, and hence a global minimum since f is quadratic. The normal equations can be written as $(b-Ax)^TA = 0$, which shows that the residual $b - Ax$ is orthogonal to range(A).

19.2. $A = Q\,[\sim]$. The computed upper triangular QR factor $\widehat{R}$ satisfies $A + \Delta A = Q\begin{bmatrix}\widehat{R}\\ 0\end{bmatrix}$ with $|\Delta A| \le mn\gamma_{cm}G_1|A|$ and $\|G_1\|_F = 1$, by Theorem 18.4. By Lemma 18.3 the computed transformed right-hand side satisfies $\begin{bmatrix}\widehat{c}\\ \widehat{d}\end{bmatrix} = Q^T(b + \Delta b)$, with $|\Delta b| \le mn\gamma_{cm}G_1\,|b|$.

By Theorem 8.5, the computed solution $\widehat{x}$ to the triangular system $\widehat{R}x = \widehat{c}$ satisfies

$$(\widehat{R} + \Delta R)\widehat{x} = \widehat{c}, \qquad |\Delta R| \le \gamma_n|\widehat{R}|.$$

So $\widehat{x}$ exactly solves the LS problem

$$\begin{bmatrix}\widehat{R} + \Delta R\\ 0\end{bmatrix} x \approx \begin{bmatrix}\widehat{c}\\ \widehat{d}\end{bmatrix},$$

which is equivalent to the LS problem, on premultiplying by Q,

$$(A + \overline{\Delta A})x := \left(A + \Delta A + Q\begin{bmatrix}\Delta R\\ 0\end{bmatrix}\right)x \approx b + \Delta b.$$

We have

$$\begin{aligned}
|\overline{\Delta A}| &\le mn\gamma_{cm}G_1|A| + \gamma_n|Q|\begin{bmatrix}|\widehat{R}|\\ 0\end{bmatrix}\\
&= mn\gamma_{cm}G_1|A| + \gamma_n|Q||Q^T(A + \Delta A)|\\
&\le mn\gamma_{cm}G_1|A| + \gamma_n|Q||Q^T|(I + mn\gamma_{cm}G_1)|A| =: mn\gamma'_{cm}G|A|,
\end{aligned}$$

where $G \ge G_1$ and $\|G\|_F = 1$. The normwise bounds are proved similarly.

19.3. A straightforward verification.

19.4. Notice that

$$\|AX - I_m\|_F^2 = \|A[x_1, \ldots, x_m] - [e_1, \ldots, e_m]\|_F^2 = \sum_{i=1}^{m} \|Ax_i - e_i\|_2^2,$$

which is minimized if $\|Ax_i - e_i\|_2$ is minimized for $i = 1\colon m$. Thus we have m independent LS problems. The solution is $x_i = A^+ e_i$, $i = 1\colon m$, that is, $X = A^+ I_m = A^+$. This is the unique solution only if A has full rank, but it is always the unique minimum Frobenius norm solution.

19.5. By Theorem 18.12 there is an orthonormal matrix $[W_1, w_{n+1}] \in \mathbb{R}^{m \times (n+1)}$ such that

$$[\,A + \Delta A \quad b + \Delta b\,] = [\,W_1 \quad w_{n+1}\,] \begin{bmatrix} \widehat{R} & \widehat{z} \\ 0 & \widehat{\rho} \end{bmatrix},$$

with $|\Delta A| \le c_{m,n} u G |A|$, $|\Delta b| \le c_{m,n} u G |b|$, $\|G\|_F = 1$. The computed solution $\widehat{x}$ to $\widehat{R}x = \widehat{z}$ satisfies

$$(\widehat{R} + \Delta R)\widehat{x} = \widehat{z}, \qquad |\Delta R| \le \gamma_n |\widehat{R}|.$$

Therefore $\widehat{x}$ exactly solves the LS problem

$$\begin{bmatrix} \widehat{R} + \Delta R \\ 0 \end{bmatrix} x \approx \begin{bmatrix} \widehat{z} \\ \widehat{\rho} \end{bmatrix},$$

which, on premultiplying by $[W_1, w_{n+1}]$, is equivalent to the LS problem

$$W_1(\widehat{R} + \Delta \widehat{R})x \approx W_1 \widehat{z} + w_{n+1}\widehat{\rho},$$

$$(A + \overline{\Delta A})x := (A + \Delta A + W_1 \Delta R)x \approx b + \Delta b.$$

We have

$$\begin{aligned} |\overline{\Delta A}| &\le c_{m,n} u G |A| + \gamma_n |W_1||\widehat{R}| \\ &= c_{m,n} u G |A| + \gamma_n |W_1||W_1^T (A + \Delta A)| \\ &\le c_{m,n} u G |A| + \gamma_n |W_1||W_1^T|(I + c_{m,n} u G)|A| \\ &= c'_{m,n} u \overline{G} |A|, \end{aligned}$$

where $\overline{G} \ge G$ and $\|\overline{G}\|_F = 1$. Normwise bounds on $\overline{\Delta A}$ and Δb follow similarly.

19.6. Subtracting $\widehat{r}^T\widehat{r} = b^T\widehat{r} - \widehat{x}^T A^T \widehat{r}$ from $r^T r = r^T b$, we have

$$\begin{aligned} r^T r - \widehat{r}^T \widehat{r} &= b^T (r - \widehat{r}) + \widehat{x}^T A^T \widehat{r} \\ &= b^T A(\widehat{x} - x) + \widehat{x}^T (A^T b - A^T A \widehat{x}) \\ &= x^T A^T A(\widehat{x} - x) + \widehat{x}^T (A^T b - A^T A \widehat{x}) \\ &= (x - \widehat{x})^T (A^T A \widehat{x} - A^T b) \\ &= (x - \widehat{x})^T (\Delta c - \Delta A \widehat{x}) \qquad \text{by (19.12).} \end{aligned}$$

Taking norms gives the result.

19.7. By constructing a block LU factorization it is straightforward to show that

$$\det(C(\alpha) - \lambda I) = (\alpha - \lambda)^{m-n} \det(A^TA + \lambda(\alpha - \lambda)).$$

Hence the eigenvalues of $C(\alpha)$ are $\lambda = \alpha$ ($m - n$ times) together with the solutions of $\lambda(\alpha - \lambda) = -\sigma_i^2$, namely, $\lambda = \alpha/2 \pm (\alpha^2/4 + \sigma_i^2)^{1/2}$, $i = 1{:}n$.

Now

$$\sigma_{\min}(C(\alpha)) = \min\left\{ \alpha, \left(\frac{\alpha^2}{4} + \sigma_n^2\right)^{1/2} - \frac{\alpha}{2} \right\},$$

so the minimum condition number must occur when $\alpha = (\alpha^2/4+\sigma_n^2)^{1/2} - \alpha/2$. This gives $\alpha = \sigma_n/\sqrt{2}$, for which

$$\kappa_2(C(\alpha)) = \frac{1}{2} + \left(\frac{1}{4} + 2\kappa_2(A)^2\right)^{1/2}.$$

The lower bound for the maximum is achieved for$\alpha = \sigma_1/\sqrt{2}$.

19.8. Let $y = 0$ and $\|(A + DA)y - b\|_2 = \min$. If $b = 0$ then we can take $AA= O$. Otherwise, the normal equations tell us that $(A+ \Delta A)^Tb = 0$, so $\|\Delta A\|_2 \ge \|A^Tb\|_2/\|b\|_2$. This lower bound is achieved, and the normal equations satisfied, for $\Delta A^T = -(A^Tb)b^T/b^Tb$. Hence

$$\eta_F(0) = \begin{cases} \|A^Tb\|_2/\|b\|_2, & b \ne 0, \\ 0, & b = 0. \end{cases}$$

19.9. For the case $\lambda_* < 0$ we have

$$\begin{aligned}
\eta_F(y) &= \lambda_{\min}\left(AA^T - \mu\frac{rr^T}{\|y\|_2^2} + \mu\frac{\|r\|_2^2}{\|y\|_2^2}I\right)^{1/2} \\
&= \lambda_{\min}\left(AA^T + \mu\frac{\|r\|_2^2}{\|y\|_2^2}\left(I - \frac{rr^T}{\|r\|_2^2}\right)\right)^{1/2} \\
&= \lambda_{\min}\left(AA^T + \mu\frac{\|r\|_2^2}{\|y\|_2^2}\left(I - \frac{rr^T}{\|r\|_2^2}\right)^2\right)^{1/2} \\
&= \sigma_{\min}\left(\left[A \quad \sqrt{\mu}\frac{\|r\|_2}{\|y\|_2}\left(I - \frac{rr^T}{\|r\|_2^2}\right)\right]\right).
\end{aligned}$$

20.1. Setting the gradient of (20.13) to zero gives $A^TAx - A^Tb + c = 0$, which can be written as $y = b - Ax$, $A^Ty = c$, which is (20.12). The Lagrangian for (20.14) is $L(y, x) = ½(y - b)^T(y - b) + x^T(A^Ty - c)$. $\nabla_yL(y, x) = y - b + Ax$, and $\nabla_xL(y, x) = A^Ty - c$. Setting the gradients to zero gives the system (20.12).

21.1. The modified version has the same flop count as the original version.

21.4. The summation gives

$$\sum_{j=0}^{n} x^j \sum_{i=0}^{n} w_{ij} = \sum_{i=0}^{n} l_i(x) \equiv 1,$$

which implies the desired equality. It follows that all columns of V^{-1} except the first must have both positive and negative entries. In particular, $V^{-1} \geq 0$ is not possible. The elements of V^{-1} sum to 1, independently of the points α_i (see also Problem 13.7).

21.5. We have $U(i,:)T(\alpha_o, \ldots, \alpha_n) = W(i,:)V(\alpha_o, \ldots, \alpha_n)$- But $T = LV$, where L has the form illustrated for n = 5 by

$$L = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ -1 & 0 & 2 & & \\ 0 & -3 & 0 & 4 & \\ 1 & 0 & -8 & 0 & 8 \end{bmatrix}$$

and $Le = e$, so $U(i,:)L = W(i,:)$, or $U(i,:)^T = L^{-T}W(i,:)^T$. But $L^{-T} \geq 0$ by the given x^n formula, so $\|L^{-T}\|_1 = \|L^{-1}\|_\infty = \|L^{-1}e\|_\infty = \|e\|_\infty = 1$, hence $\|U(i,:)\|_1 \leq \|W(i,:)\|_1$.

As an aside, we can evaluate $\|L\|_\infty$ *as*

$$\|L\|_\infty = |T_n(\sqrt{-1})| = \frac{1}{2}\left[(1+\sqrt{2})^n + (1-\sqrt{2})^n\right],$$

after a little trigonometric algebra.

21.7. Denote the matrix by C. For the zeros of T_{n+1} we have

$$CC^T = (n+1)\,\mathrm{diag}\left(1, \frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}\right).$$

It follows that $\|C\|_2^2 = \sqrt{n+1}$, $\|C^{-1}\|_2^2 = 2/\sqrt{n+1}$, giving the result.

Extrema of T_n: we have

$$CDC^T = \frac{n}{2}D^{-1}, \qquad D = \mathrm{diag}\left(\frac{1}{2}, 1, 1, \ldots, 1, \frac{1}{2}\right).$$

Hence $B = CD^{1/2}$ satisfies $BB^T = \frac{n}{2}D^{-1}$,and so $k_2(B) = K_2(D)^{1/2} = \sqrt{2}$. Then $k_2(C) \leq k2(D^{1/2})k_2(B) = 2$.

21.8. The increasing ordering is never produced, since the algorithm must choose α_1 to maximize $|\alpha_1 - \alpha 0|$.

21.10. The dual condition number is

$$\frac{\|\,|V^{-T}||V^T \mathrm{diag}(0,1,2,\ldots,n)a|\,\|_\infty}{\|a\|_\infty}.$$

See Higham [533, 1987] for proofs.

22.1. Consider the case where $m \le \min(n,p)$, and suppose $n = jm$ and $p = km$ for some integers j and k. Then the multiplication of the two matrices can be split into m x m blocks:

$$AB = [A_1 \quad A_2 \quad \dots \quad A_j] \begin{bmatrix} B_{11} & \dots & B_{1k} \\ \vdots & & \vdots \\ B_{j1} & \dots & B_{jk} \end{bmatrix},$$

which involves a total of jk multiplications of m x m matrices, each involving $O(m^\alpha)$ operations. Thus the total number of operations is $O(jkm^\alpha)$, or $O(m^{\alpha-2}np)$, as required, and we can show similar results for the cases when n and p are the smallest dimensions.

22.2. $\frac{1}{2}n^3 + n^2$ multiplications and $\frac{3}{2}n^3 + 2n(n-1)$ additions.

22.3. For large $n = 2^k$, $S_n(8)/S_n(n) \approx 1.96 \times (7/8)^k$ and $S_n(1)/S_n(8) \approx 1.79$. The ratio $S_n(8)/S_n(n)$ measures the best possible reduction in the amount of arithmetic by using Strassen's method in place of conventional multiplication. The ratio $S_n(1)/S_n(8)$ measures how much more arithmetic is used by recurring down to the scalar level instead of stopping once the optimal dimension n_0 is reached. Of course, the efficiency of a practical code for Strassen's method also depends on the various non-floating-point operations.

22.5. Apart from the differences in stability, the key difference is that Winograd's formula relies on commutativity and so cannot be generalized to matrices.

22.7. Some brief comments are given by Douglas, Heroux, Slishman, and Smith [317, 1994].

22.9. The inverse is

$$\begin{bmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix}.$$

Hence we can form AB by inverting a matrix of thrice the dimension. This result is not of practical value, but it is useful for computational complexity analysis.

24.2. With $n = 3$ and almost any starting data, the backward error can easily be made of order 1, showing that the method is unstable. However, the backward error is found to be roughly of order $\kappa_\infty(A)u$, so the method may have a backward error bound proportional to $\kappa_\infty(A)$ (this is an open question).

25.1. (b) An optimizing compiler might convert the test xp1 > 1 to x+1 > 1 and then to x > 0. (For a way to stop this conversion in Fortran, see the solution to Problem 25.3.) Then the code would compute a number of order $2^{e_{\min}}$ instead of a number of order 2^{-t}.

25.3. The algorithm is based on the fact that the positive integers that can be exactly represented are 1,2, ..., β^t and

$$\beta^t + \beta, \beta^t + 2\beta, \beta^t + 3\beta, \dots, \beta^{t+1}, \beta^{t+1} + \beta^2, \dots.$$

In the interval $[\beta^t, \beta^{t+1}]$ the floating point numbers are spaced β apart. This interval must contain a power of 2, $a = 2^k$. The first while loop finds such an a (or, rather, the floating point representation of such an a) by testing successive powers 2^i to see if both 2^i and $2^i + 1$ are representable. The next while loop adds successive powers of 2 until the next floating point number is found; on subtracting a the base β is produced. Finally t is determined as the smallest power of β for which the distance to the next floating point number exceeds 1.

The routine can fail for at least two reasons. First, an optimizing compiler might simplify the test while (a+l) - a == 1 to while 1 == 1, thus altering the meaning of the program. Second, in the same test the result (a+l) might be held in an extra length register and the subtraction done in extra precision. The computation would then reflect this higher precision and not the intended one. We could try to overcome this problem by saving the result a+l in a variable, but an optimizing compiler might undo this effort by dispensing with the variable and storing the result in a register. In Fortran, the compiler's unwanted efforts can be nullified by a test of the form while foo (a+l) - a == 1, where foo is a function that simply returns its argument. The problems caused by the use of extra length registers were discussed by Gentleman and Marovich [437, 1974]; see also Miller [759, 1984, §2.2].

25.4. (This description is based on Schreiber [903, 1989].) The random number generator in matgen repeats after 16384 numbers [668, 1981, p. 19]. The dimension n = 512 divides the period of the generator (16384 = 512 x 32), with the effect that the first 32 columns of the matrix are repeated 16 times (512 = 32 x 16), so the matrix has this structure:

```
B = rand(512,32);
A = [B, B, B, B, B, B, B, B, B, B, B, B, B, B, B, B];
```

Now consider the first 32 steps of Gaussian elimination. We apply 32 transformations to A that have the effect, in exact arithmetic, of making B upper trapezoidal. In floating point arithmetic, they leave a residue of small numbers (about $u \approx 10^{-7}$ in size) in rows 33 onward. Because of the structure of the matrix, identical small numbers occur in each of the 15 blocks of A **to** the right of the first. Thus, the remaining (512 – 32) x (512 – 32) submatrix has the same block structure as A (but with 15 block columns). Hence this process repeats every 32 steps:

after 32 steps the elements drop to $\approx 10^{-7}$;
after 64 steps the elements drop to $\approx 10^{-14}$;
after 96 steps the elements drop to $\approx 10^{-21}$;
after 128 steps the elements drop to $\approx 10^{-28}$;
after 160 steps the elements drop to $\approx 10^{-35}$;
after 192 steps the elements would drop to $\approx 10^{-42}$,

but that is less than the under flow threshold. The actual pivots do not exactly match the analysis, which is probably due to rank deficiency of one of the submatrices generated. Also, underflow occurs earlier than predicted, apparently because two small numbers (both $O(10^{-21}))$ are multiplied in a saxpy operation.

25.5. Let s_i and t_t denote the values of s and t at the start of the ith stage of the algorithm. Then

$$\sum_{k=1}^{i-1} x_k^2 = t_i^2 s_i. \tag{A.15}$$

If $|x_i| > t_i$ then the algorithm uses the relation

$$\sum_{k=1}^{i} x_k^2 = t_i^2 s_i + x_i^2 = x_i^2\big((t/x_i)^2 s_i + 1\big),$$

so with $s_{i+1} = (t_i/x_i)^2 s_i + 1$ and $t_{i+1} = |x_i|$, (A.15) continues to hold for the next value of i. The same is true trivially in the case $|x_i| \le t_i$.

This is a one-pass algorithm using n divisions that avoids overflow except, possibly, on the final stage in forming $t\sqrt{s}$, which can overflow only if $\|x\|_2$ does.

25.7. (a) The matrix $A := I + Y^2 = I - Y^TY$ is symmetric positive semidefinite since $\|Y\|_2 \le 1$, hence it has a (unique) symmetric positive semidefinite square root X satisfying $X^2 = A = I + Y^2$. The square root X is a polynomial in A *(see,* for example, Higham [532, 1987]) and hence a polynomial in Y; therefore X and Y commute, which implies that $(X+Y)^T(X+Y) = (X-Y)(X+Y) = X^2 - Y^2 = I$, as required.

25.8. $|x|/3$ could underflow to zero, or become unnormalized, when $\sqrt{|x|}/\sqrt{3}$ does not.

Appendix B

Singular Value Decomposition, *M*-Matrices

The singular value decomposition is a matrix factorization which can produce approximations to large arrays.
Cryptanalysts is the task of breaking coded messages.
In this paper, we present an unusual merger of the two in which the singular value decomposition may aid the cryptanalyst in discovering vowels and consonants in messages coded in certain variations of simple substitution ciphers.

— CLEVE B. MOLER and DONALD MORRISON,
Singular Value Analysis of Cryptograms (1983)

If $A \in Z^{n\times n}$ $[a_{ij} \leq 0$ *for all* $i \neq j]$
then each of the following 50 conditions is equivalent to the statement:
*“**A** is a nonsingular M-matrix”.*

— ABRAHAM BERMAN and ROBERT J. PLEMMONS,
Nonnegative Matrices in the Mathematical Sciences (1994)

In this appendix we define the singular value decomposition and M-matrices.

B.1. Singular Value Decomposition

Any matrix $A \in \mathbb{C}^{m \times n}$ has a *singular value decomposition* (SVD)

$$A = U\Sigma V^*, \quad \Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p) \in \mathbb{C}^{m\times n}, \quad p = \min(m, n),$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ and $U \in \mathbb{C}^{m\times m}$, $V \in \mathbb{C}^{n\times n}$ are both unitary. The σ_i are the *singular values* of A and the columns of U and V are the left and *right singular vectors* of A, respectively.

For more details on the SVD see Golub and Van Loan [470, 1989, pp. 71-73], Stewart and Sun [954, 1990, pp. 30-34], and Horn and Johnson [580, 1985, §7.3], [581, 1991, §3.1]. The history of the SVD is described by Stewart [950, 1993] and Horn and Johnson [581, 1991, \$3.0].

B . 2 . *M*-Matrices

A matrix $A \in \mathbb{R}^{n\times n}$ is an *M*-matrix if $a_{ij} \leq 0$ for all $i \neq j$ and all the eigenvalues of A have nonnegative real part. This is one of many equivalent definitions [94, 1994, Chap. 6].

An *M*-matrix may be singular. A particularly useful characterization of a nonsingular *M*-matrix is a nonsingular matrix $A \in \mathbb{R}^{n\times n}$ for which $a_{ij} \leq 0$ for all $i \neq j$, and A^{-1} has nonnegative elements (written as $A^{-1} \geq 0$).

For more information on ill-matrices see Berman and Plemmons [94, 1994] and Horn and Johnson [581, 1991, §2.5].

Appendix C

Acquiring Software

Caveat receptor . . .
Anything free comes with no guarantee!
— JACK DONGARRA and ERIC GROSSE, *Netlib mail header*

In this appendix we provide information on how to acquire software mentioned in the book. First, we describe some basic aspects of the Internet.

C. 1. Internet

A huge variety of information and software is available over the Internet, the worldwide combination of interconnected computer networks. The location of a particular object is specified by a URL, which stands for "(Uniform Resource Locator". Examples of URLS are

```
http://www.netlib.erg/index.html
ftp://ftp.netlib.   org
```

The first example specifies a World Wide Web server (http = hypertext transfer protocol) together with a file in hypertext format (html = hypertext markup language), while the second specifies an anonymous ftp site. In any URL, the site address may, optionally, be followed by a filename that specifies a particular file. For more details about the Internet see on-line information, or one of the many books on the subject, such as Krol [674, 1994].

C.2. Netlib

Netlib is a repository of freely available mathematical software, documents, and databases of interest to the scientific computing community [316, 1987], [151, 1994]. It includes

- research codes,
- golden oldies (classic programs that are not available in standard libraries),
- the collected algorithms of the ACM,
- program libraries such as EISPACK, LINPACK, LAPACK, and MINPACK
- back issues of NA-Digest, a weekly digest for the numerical analysis community,
- databases of conferences and performance data for a wide variety of machines.

Netlib also enables the user to download technical reports from certain institutions, to download software and errata for textbooks, and to search "the SIAM membership list and a "white pages" database.

Netlib can be accessed in several ways.

- Over the World Wide Web. The URL is
  `http://www.netlib. erg/index.html`
- Via Xnetlib, an X Windows application that provides interactive’ access to netlib.
- By anonymous ftp. The URL is
  `ftp://ftp.netlib.org`
- By electronic mail. For an introduction and master index, send a one-line email message as follows:

  ```
  mail netlib@ornl. gov
  send index
  ```

Netlib is mirrored at various sites throughout the world.

C.3. MATLAB

MATLAB is a commercial program sold by The MathWorks, Inc. It runs on a variety of platforms. The MathWorks maintains a collection of user-contributed M-files, which is accessible over the Internet.

For information contact

The MathWorks, Inc.
24 Prime Park Way
Natick, MA 01760-1500
USA

Tel: 5086531415
Fax: 5086532997

email: info@mathworks.com
URL: `ftp://ftp.mathworks.com`
URL: `http://www.mathworks.com`

C.4. NAG Library and FTN90 Compiler

The Numerical Algorithms Group (NAG) produces a variety of software products. Relevant to this book are the FTN90 compiler and the NAG Library, a large numerical program library available in several programming languages.

For information contact

NAG Ltd.
Wilkinson House

Jordan Hill Road
Oxford, OX2 8DR
U K

Tel: +44 1865 511245
Fax: +44 1865 310139

email: infodesk@nag.co.uk
URL: http://www.nag.co.uk:70/

NAG has subsidiaries and distributors, whose addresses can be obtained from the above sources.

Appendix D
Program Libraries

Since the programming is likely to be the
main bottleneck in the use of an electronic computer
we have given a good deal of thought to the
preparation of standard routines of considerable generality for the
more important processes involved in computation.
By this means we hope to reduce the time taken
to code up large-scale computing problems,
by building them up, as it were,
from prefabricated units.

— J. H. WILKINSON, *The Automatic Computing Engine at the National Physical Laboratory* (1948)

In spite of the self-contained nature of the linear algebra field,
experience has shown that even here
the preparation of a fully tested set of algorithms
is a far greater task than had been anticipated.

— J. H. WILKINSON and C. REINSCH, *Handbook for Automatic Computation: Linear Algebra* (1971)

In this appendix we briefly describe some of the freely available program libraries that have been mentioned in this book. These packages are all available from netlib (see §C.2).

D.1. Basic Linear Algebra Subprograms

The Basic Linear Algebra Subprograms (BLAS) are sets of Fortran primitives for matrix and vector operations. They cover all the common operations in linear algebra. There are three levels, corresponding to the types of object operated upon. In the examples below, x and y are vectors, A, B, C are rectangular matrices, and T is a square triangular matrix. Names of BLAS routines are given in parentheses. The leading "x" denotes the Fortran data type, whose possible values are some or all of

`S` `real`
`D` `double precision`
`C` `complex`
`Z` `complex* 16, or double complex`

Level 1: [694, 1979] *Vector operations*. Inner product: x^Ty (xdot); $y \leftarrow \alpha x + y$ (xAXPY); vector 2-norm $(y^Ty)^{1/2}$ (xNRM2); swap vectors $x \leftrightarrow y$ (xSWAP); scale a vector $x \leftarrow \alpha x$ (xSCAL); and other operations.

Level 2: [313, 1988], [314, 1988] *Matrix-vector operations*. Matrix times vector (gaxpy): $y \leftarrow aAx + by$ (xGEMV); rank-1 update: $A \leftarrow A + \alpha xy^T$ (xGER); triangular solve: $x \leftarrow T^{-1}x$ (xTRSV); and variations on these.

Level 3: [308, 1990], [309, 1990] *Matrix-matrix operations*. Matrix multiplication: $C \leftarrow aAB + bC$ (xGEMM); multiple right-hand side triangular solve: $A \leftarrow aT^{-1}A$ (xTRSM); rank-r and rank-$2r$ updates of a symmetric matrix (xSYRK, xSYR2K); and variations on these.

The BLAS are intended to be called in innermost loops of linear algebra codes. Usually, most of the computation in a code that uses BLAS calls is done inside these calls. LINPACK [307, 1979] uses the level-1 BLAS throughout (model Fortran implementations of the level-1 BLAS are listed in [307, 1979, App. D]). LAPACK [17, 1995] exploits all three levels, using the highest possible level at all times.

Each set of BLAS comprises a set of *subprogram specifications* only. The specifications define the parameters to each routine and state what the routine must do, but not how it must do it. Thus the implementor has freedom over the precise implementation details (loop orderings, block algorithms, special code for special cases) and even the method (fast versus conventional matrix multiply), but the implementation is required to be numerically stable, and code that tests the numerical stability is provided with the model implementations [309, 1990], [314, 1988].

For more details on the BLAS and the advantages of using them, see the defining papers listed above, or, for example, [315, 1991] or [470, 1989, Chap. 1].

D.2. EISPACK

EISPACK is a collection of Fortran 66 subroutines for computing eigenvalues and eigenvectors of matrices [925, 1976], [415, 1977]. It contains 58 subroutines and 13 drivers. The subroutines are the basic building blocks for eigensystem computations; they perform such tasks as reduction to Hessenberg form, computation of some or all of the eigenvalues/vectors, and back transformations, for various types of matrix (real, complex, symmetric, banded, etc.). The driver subroutines provide easy access to many of EISPACK's capabilities; they call from one to five other EISPACK subroutines to do the computations.

EISPACK is primarily based on Algol 60 procedures developed in the 1960s by 19 different authors and published in the journal *Numerische Mathematik.* An edited collection of these papers was published in the *Handbook for Automatic Computation* [1102, 1971].

D.3. LINPACK

LINPACK is a collection of Fortran 66 subroutines that analyse and solve linear equations and linear least squares problems [307, 1979]. The package solves linear systems whose matrices are general, banded, symmetric indefinite, symmetric positive definite, triangular, or tridiagonal. In addition, the package computes the QR and singular value decompositions and applies them to least squares problems. All the LINPACK routines use calls to the level-l BLAS in the innermost loops; thus most of the floating point arithmetic in LINPACK is done within the level-l BLAS.

D.4. LAPACK

LAPACK [17, 1995] was released on February 29, 1992. As the announcement stated, "LAPACK is a transportable library of Fortran 77 subroutines for solving the most common problems in numerical linear algebra: systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems. It has been designed to be efficient on a wide range of modern high-performance computers."

LAPACK has been developed over a period that began in 1987 by a team of 11 numerical analysts in the UK and the USA. LAPACK can be regarded as a successor to LINPACK and EISPACK; it has virtually all their capabilities and much more besides. LAPACK improves on LINPACK and EISPACK in four main respects: speed, accuracy, robustness, and functionality. It was designed at the outset to exploit the level-3 BLAS.

Development of LAPACK continues under the auspices of two follow-on projects, LAPACK 2 and ScaLAPACK. An object-oriented C++ extension to LAPACK has been produced, called LAPACK++ [311, 1995]. CLAPACK is a C version of LAPACK, converted from the original Fortran version using the `f2c` converter [367, 1990]. ScaLAPACK comprises a subset of LAPACK routines redesigned for distributed memory parallel machines [206, 1992], [205, 1994]. Other work includes developing codes that take advantage of the careful rounding and exception handling of IEEE arithmetic [298, 1994]. For more details of all these topics see [17, 1995].

LAPACK undergoes regular updates, which are announced on the electronic newsletter NA-Digest. At the time of writing, the current release is version 2.0, dated September 30, 1994, and the package contains over 1000 routines and over 735,000 lines of Fortran 77 code, including testing and timing code.

Mark 16 onward of the NAG Fortran 77 Library contains much of LAPACK in Chapters F07 and F08.

D.4.1. Structure of LAPACK

The LAPACK routines can be divided into three classes.

The *drivers* solve a complete problem. The *simple* drivers have a minimal specification, while the *expert* drivers have additional capabilities of interest to the sophisticated user. The *computational* routines perform individual tasks such as computing a factorization or reducing a matrix to condensed form; they are called by the drivers. The *auxiliary* routines perform relatively low-level operations such as unblocked factorization, estimating or computing matrix norms, and solving a triangular system with scaling to prevent overflow.

The driver and computational routines have names of the form xyyzzz. The first letter specifies the data type, which is one of S, D, C, and Z. The second two letters refer to the type of matrix. A partial list of types is as follows (there are 27 types in all):

BD	bidiagonal
GB	general band
GE	general
GT	general tridiagonal
HS	upper Hessenberg
OR	(real) orthogonal
PO	symmetric or Hermitian positive definite
PT	symmetric or Hermitian positive definite tridiagonal
SB	(real) symmetric band
ST	(real) symmetric tridiagonal
SY	symmetric
TR	triangular (or (quasi-triangular)

The last three characters specify the computation performed.

TRF	factorize
TRS	solve a (multiple right-hand side) linear system using the factorization
CON	estimate $1/k_1(A)$ (or compute it exactly when A is tridiagonal and symmetric positive definite or Hermitian positive definite)
RFS	apply fixed precision iterative refinement and compute the componentwise relative backward error and a forward error bound
TRI	use the factorization to compute A^{-1}
EQU	compute factors to equilibrate the matrix

The auxiliary routines follow a similar naming convention, with most of them having yy = LA.

Appendix E
The Test Matrix Toolbox

The Test Matrix Toolbox is a collection of MATLAB M-files containing test matrices, routines for visualizing matrices, routines for direct search optimization, and miscellaneous routines that provide useful additions to MATLAB'S existing set of functions. There are over 50 parametrized test matrices, which are mostly square, dense, nonrandom, and of arbitrary dimension. The test matrices include ones with known inverses or known eigenvalues; ill-conditioned or rank deficient matrices; and symmetric, positive definite, orthogonal, defective, involuntary, and totally positive matrices. The visualization routines display surface plots of a matrix and its (pseudo-) inverse, the field of values, Gershgorin disks, and two- and three-dimensional views of pseudospectra. The direct search optimization routines implement the alternating directions method, the multidirectional search method, and the Nelder–Mead simplex method (which are described in $24.2).

Among the miscellaneous routines are one for rounding matrix elements to a specified number of bits and ones that implement the classical and modified Gram-Schmidt methods and LU factorization without pivoting and with complete pivoting.

The Test Matrix Toolbox was originally released in 1989, and a version was published as ACM Algorithm 694 [548, 1991]. The current version, version 3.0, is described in the manual [561, 1995], which should be consulted for further details.

The Test Matrix Toolbox is available by anonymous ftp from The MathWorks; the URL is

`ftp://ftp.mathworks.com/pub/contrib/linalg/testmatrix`

The manual [561, 1995] is testmatrix.ps in the same location.

We summarize the contents of the toolbox in the following tables, which list the M-files by category, with short descriptions.

Demonstration	
`tmtdemo`	Demonstration of Test Matrix Toolbox.

Test Matrices, A–K	
`augment`	Augmented system matrix.
`cauchy`	Cauchy matrix.
`chebspec`	Chebyshev spectral differentiation matrix.
`chebvand`	Vandermonde-like matrix for the Chebyshev polynomials.
`chow`	Chow matrix—a singular Toeplitz lower Hessenberg matrix.
`circul`	Circulant matrix.
`clement`	Clement matrix—tridiagonal with zero diagonal entries.
`compan`	Companion matrix.
`condex`	"Counterexamples" to matrix condition number estimators.
`cycol`	Matrix whose columns repeat cyclically.
`dingdong`	Dingdong matrix—a symmetric Hankel matrix.
`dorr`	Dorr matrix—diagonally dominant, ill conditioned, tridiagonal.
`dramadah`	A (O, 1) matrix whose inverse has large integer entries.
`fiedler`	Fiedler matrix—symmetric.
`forsythe`	Forsythe matrix—a perturbed Jordan block.
`frank`	Frank matrix—ill conditioned eigenvalues.
`gallery`	Famous, and not so famous, test matrices.
`gearm`	Gear matrix.
`gfpp`	Matrix giving maximal growth factor for Gaussian elimination with partial pivoting.
`grcar`	Grcar matrix—a Toeplitz matrix with sensitive eigenvalues.
`hadamard`	Hadamard matrix.
`hanowa`	A matrix whose eigenvalues lie on a vertical line in the complex plane.
`hilb`	Hilbert matrix.
`invhess`	Inverse of an upper Hessenberg matrix.
`invol`	An involutory matrix.
`ipjfact`	A Hankel matrix with factorial elements.
`jordbloc`	Jordan block.
`kahan`	Kahan matrix—upper trapezoidal.
`kms`	Kac–Murdock–Szego Toeplitz matrix.
`krylov`	Krylov matrix.

Test Matrices. L–Z	
`lauchli`	Lauchli matrix—rectangular.
`lehmer`	Lehmer matrix-symmetric positive definite.
`lesp`	A tridiagonal matrix with real, sensitive eigenvalues.
`lotkin`	Lotkin matrix.
`makejcf`	A matrix with given Jordan canonical form.
`minij`	Symmetric positive definite matrix min(i,j).
`moler`	Moler matrix-symmetric positive definite.
`neumann`	Singular matrix from the discrete Neumann problem (sparse).
`ohess`	Random, orthogonal upper Hessenberg matrix.
`orthog`	Orthogonal and nearly orthogonal matrices.
`parter`	Parter matrix—a Toeplitz matrix with singular values near p.
`pascal`	Pascal matrix.
`pdtoep`	Symmetric positive definite Toeplitz matrix.
`pei`	Pei matrix.
`pentoep`	Pentadiagonal Toeplitz matrix (sparse).
`poisson`	Block tridiagonal matrix from Poisson's equation (sparse).
`prolate`	Prolate matrix—symmetric, ill-conditioned Toeplitz matrix.
`rando`	Random matrix with elements – 1, 0, or 1.
`randsvd`	Random matrix with pre-assigned singular values.
`redheff`	A (0,1) matrix of Redheffer associated with the Riemann hypothesis.
`riemann`	A matrix associated with the Riemann hypothesis.
`rschur`	An upper quasi-triangular matrix.
`smoke`	Smoke matrix-complex, with a "smoke ring" pseudospectrum.
`tridiag`	Tridiagonal matrix (sparse).
`triw`	Upper triangular matrix discussed by Wilkinson and others.
`vend`	Vandermonde matrix.
`wathen`	Wathen matrix—a finite element matrix (sparse, random entries).
`wilk`	Various specific matrices devised/discussed by Wilkinson.

Visualization	
`fv`	Field of values (or numerical range).
`gersh`	Gershgorin disks.
`ps`	Dot plot of a pseudospectrum.
`pscont`	Contours and colour pictures of pseudospectra.
`see`	Pictures of a matrix and its (pseudo-) inverse.

Decompositions and Factorization	
`cgs`	Classical Gram–Schmidt QR factorization.
`cholp`	Cholesky factorization with pivoting of a positive semidefinite matrix.
`cod`	Complete orthogonal decomposition.
`diagpiv`	Diagonal pivoting factorization with partial pivoting.
`ge`	Gaussian elimination without pivoting.
`gecp`	Gaussian elimination with complete pivoting.
`gj`	Gauss-Jordan elimination to solve $Ax = b$.
`mgs`	Modified Gram–Schmidt QR factorization.
`poldec`	Polar decomposition.
`signm`	Matrix sign decomposition.

Direct Search Optimization	
`adsmax`	Alternating directions direct-search method.
`mdsmax`	Multidirectional search method for direct search optimization.
`mmsmax`	Nelder–Mead simplex method for direct search optimization.

Miscellaneous	
`bandred`	Band reduction by two-sided unitary transformations.
`chop`	Round matrix elements.
`comp`	Comparison matrices.
`cond`	Matrix condition number in 1, 2, Frobenius, or ∞-norm.
`cpltaxes`	Determine suitable axis for plot of complex vector.
`dual`	Dual vector with respect to Holder p-norm.
`eigsens`	Eigenvalue condition numbers.
`house`	Householder matrix.
`matrix`	Test Matrix Toolbox information and matrix access by number.
`matsignt`	Matrix sign function of a triangular matrix.
`pnorm`	Estimate of matrix p-norm ($1 \le p \le \infty$).
`qmult`	Pre-multiply by random orthogonal matrix.
`rq`	Rayleigh quotient.
`seqa`	Additive sequence.
`seqcheb`	Sequence of points related to Chebyshev polynomials.
`seqm`	Multiplicative sequence.
`show`	Display signs of matrix elements.
`skewpart`	Skew-symmetric (skew-Hermitian) part.
`sparsify`	Randomly sets matrix elements to zero.
`sub`	Principal submatrix.
`symmpart`	Symmetric (Hermitian) part.
`trap2tri`	Unitary reduction of trapezoidal matrix to triangular form.

Bibliography

[1] Jan Ole Aasen. On the reduction of a symmetric matrix to tridiagonal form. *BIT,* 11:233-242, 1971.

[2] Nabih N. Abdelmalek. Round off error analysis for Gram-Schmidt method and solution of linear least squares problems. *BIT,* 11 :345–368, 1971.

[3] *ACM Turing Award Lectures: The First Twenty Years,* 1966–1985. Addison-Wesley, Reading, MA, USA, 1987. xviii+483 pp. ISBN 0-201-54885-2.

[4] Forman S. Acton. *Numerical Methods That Work.* Harper and Row, New York, 1970. xviii+541 pp. Reprinted by Mathematical Association of America, Washington, DC, with new preface and additional problems, 1990. ISBN 0-88385-450-3.

[5] Duane A. Adams. A stopping criterion for polynomial root finding. *Comm. ACM,* 10:655-658, 1967.

[6] Vijay B. Aggarwal and James W. Burgmeier. A round-off error model with applications to arithmetic expressions. *SIAM J. Comput.,* 8(1):60-72, 1979.

[7] Alan A. Ahac, John J. Buoni, and D. D. Olesky. Stable *LU* factorization of H-matrices. *Linear Algebra Appl.,* 99:97–110, 1988.

[8] J. H. Ahlberg and E. N. Nilson. Convergence properties of the spline fit. *J. Soc. Indust. Appl. Math.,* 11(1):95-104, 1963.

[9] Paul Halmos by parts (interviews by Donald J. Albers). In *Paul IIalmos: Celebrating* 50 *Years of Mathematics,* John H. Ewing and F. W. Gehring, editors, Springer-Verlag, Berlin, 1991, pages 3–32.

[10] Göltz Alefeld and Jurgen Herzberger. *Introduction to Interval Computations.* Academic Press, New York, 1983. xviii+333 pp. ISBN 0-12-049820-0.

[11] M. Almacany, C. B. Dunham, and J. Williams. Discrete Chebyshev approximation by interpolating rationals. *IMA J. Numer. Anal.,* 4:467–477, 1984,

[12] Steven C. Althoen and Renate McLaughlin. Gauss-Jordan reduction: A brief history. *Amer. Math. Monthly,* 94(2):130-142, 1987.

[13] Fernando L. Alvarado, Alex Pothen, and Robert S. Schreiber. Highly parallel sparse triangular solution. In *Graph Theory and Sparse Matrix Computations,* J. Alan George, John R. Gilbert, and Joseph W. H. Liu, editors, volume 56 of *IMA Volumes in Mathematics and its Applications,* Springer-Verlag, New York, 1993, pages 141-158.

[14] William F. Ames. *Numerical Methods for Partial Differential Equations.* Second edition, Academic Press, New York, 1977. xiv+365 pp. ISBN 0-12-056760-1.

[15] Pierluigi Amodio and Francesca Mazzia. Backward error analysis of cyclic reduction for the solution of tridiagonal systems. *Math. Comp.,* 62(206): 601-617, 1994.

[16] Andrew A. Anda and Haesun Park. Fast plane rotations with dynamic scaling. *SIAM J. Matrix Anal. Appl.,* 15(1):162–174, 1994.

[17] E. Anderson, Z. Bai, C. H. Bischof, J. W. Demmel, J. J. Dongarra, J. J. Du Croz, A. Greenbaum, S. J. Hammarling, A. McKenney, S. Ostrouchov, and D. C. Sorensen. *LAPACK Users' Guide, Release* 2.0. Second edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xix+325 pp. ISBN 0-89871-345-5.

[18] Edward Anderson. Robust triangular solves for use in condition estimation. Technical Report CS-91-142, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, August 1991. 35 pp. LAPACK Working Note 36.

[19] T. W. Anderson. *The Statistical Analysis of Time Series.* Wiley, New York, 1971. xiv+704 pp. ISBN 0-471-02900-9.

[20] T. W. Anderson, I. Olkin, and L. G. Underhill. Generation of random orthogonal matrices. *SIAM J. Sci. Statist. Comput.,* 8(4):625+29, 1987.

[21] T. Ando. Totally positive matrices. *Linear Algebra Appl.,* 90:165-219, 1987.

[22] Anonymous. Le Commandant Cholesky. *Bulletin Géodésique,* pages 159-161, 1922. Translation by Richard W. Cottle ("Major Cholesky") appears in [721, Appendix] and in NA-Digest, Volume 90, Issue 7, 1990.

[23] Anonymous. James Wilkinson (1919-1986). *Ann. Hist. Comput.,* 9(2):205-210, 1987. From the introduction: "A series of lightly edited extracts from messages that were sent over various computer networks during the period October 5, 1986-February 13, 1987".

[24] M. Arioli, J. W. Demmel, and I. S. Duff. Solving sparse linear systems with sparse backward error. *SIAM J. Matrix Anal. Appl.,* 10(2):165–190, 1989.

[25] M. Arioli, I. S. Duff, and P. P. M. de Rijk. On the augmented system approach to sparse least-squares problems. *Numer. Math.,* 55:667-684, 1989.

[26] M. Arioli and A. Laratta. Error analysis of an algorithm for solving an underdetermined linear system. *Numer. Math.,* 46:255-268, 1985.

[27] M. Arioli and A. Laratta. Error analysis of algorithms for computing the projection of a point onto a linear manifold. *Linear Algebra Appl.,* 82:1–26, 1986.

[28] Mario Arioli, Iain S. Duff, and Daniel Ruiz. Stopping criteria for iterative solvers. *SIAM J. Matrix Anal. Appl.,* 13(1):138–144, 1992.

[29] Mario Arioli and Francesco Romani. Stability, convergence, and conditioning of stationary iterative methods of the form $x^{(i+1)} = Px^{(i)} + q$ for the solution of linear systems. *IMA J. Numer. Anal.,* 12:21–30, 1992.

[30] William F. Arnold and Alan J. Laub. Generalized eigenproblem algorithms and software for algebraic Riccati equations. *Proc. IEEE,* 72(12): 1746–1754, 1984.

[31] R. L. Ashenhurst and N. Metropolis. Error estimation in computer calculation. *Amer. Math. Monthly,* 72(2):47–58, 1965.

[32] Edgar Asplund. Inverses of matrices $\{a_{ij}\}$ which satisfy $a_{ij} = 0$ for $j > i + p$. *Math. Stand.,* 7:57-60, 1959.

[33] John V. Atanasoff. Computing machine for the solution of large systems of linear algebraic equations. Unpublished manuscript, Iowa State College, Ames, IA, USA, August 1940. Reprinted in [860, pp. 315-335].

[34] Owe Axelsson. *Iterative Solution Methods.* Cambridge University Press, 1994. xiii+654 pp. ISBN 0-521-44524-8.

[35] Ivo Babuška. Numerical stability in mathematical analysis. In *Proc. IFIP Congress,* Information Processing 68, North-Holland, Amsterdam, The Netherlands, 1969, pages 11-23.

[36] Zhaojun Bai. A collection of test matrices for large scale nonsymmetric eigenvalue problems (version 1.0). Manuscript, July 1994.

[37] Zhaojun Bai and James W. Demmel. Design of a parallel nonsymmetric eigenroutine toolbox, Part I. In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Volume* I, Richard F. Sincovec, David E. Keyes, Michael R. Leuze, Linda R. Petzold, and Daniel A. Reed, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1993, pages 391–398.

[38] Zhaojun Bai and James W. Demmel. On swapping diagonal blocks in real Schur form. *Linear Algebra Appl.,* 186:73-95, 1993.

[39] Zhaojun Bai, James W. Demmel, and Ming Gu. Inverse free parallel spectral divide and conquer algorithms for nonsymmetric eigenproblems. Computer Science Division Report UCB/CSD-94-793, University of California, Berkeley, February 1994. 34 pp.

[40] Zhaojun Bai, James W. Demmel, and Alan McKenney. On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Software,* 19(2):202–223, 1993.

[41] D. H. Bailey and H. R. P. Ferguson. A Strassen-Newton algorithm for high-speed parallelizable matrix inversion. In *Proceedings of Supercomputing '88,* IEEE Computer Society Press, New York, 1988, pages 419–424.

[42] David H. Bailey. The computation of π to 29,360,000 decimal digits using Borweins' quartically convergent algorithm. *Math. Comp.,* 50(181):283-296, 1988.

[43] David H. Bailey. Extra high speed matrix multiplication on the Cray-2. *SIAM J. Sci. Statist. Comput.,* 9(3):603-607, 1988.

[44] David H. Bailey. Algorithm 719: Multiprecision translation and execution of FORTRAN programs. *ACM Trans. Math. Software,* 19(3):288-319, 1993.

[45] David H. Bailey. A Fortran-90 based multiprecision system. Technical Report RNR-94-013, NASA Ames Research Center, Moffett Field, CA, USA, June 1994. 12 pp.

[46] David H. Bailey, Robert Krasny, and Richard Pelz. Multiple precision, multiple processor vortex sheet roll-up computation. In *Proceedings of the Sixth SIAM Conference on Parallel Processing for Scientific Computing, Volume* I, Richard F. Sincovec, David E. Keyes, Michael R. Leuze, Linda R. Petzold, and Daniel A. Reed, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1993, pages 52-56.

[47] David H. Bailey, King Lee, and Horst D. Simon. Using Strassen's algorithm to accelerate the solution of linear systems. *J. Supercomputing*, 4:357–371, 1991.

[48] David H. Bailey, Horst D. Simon, John T. Barton, and Martin J. Fouts. Floating point arithmetic in future supercomputers. *Internat. J. Supercomputer Appl.*, 3(3):86-90, 1989.

[49] J. K. Baksalary and R. Kala. The matrix equation $AX - YB = C$. *Linear Algebra Appl.*, 25:41-43, 1979.

[50] J. K. Baksalary and R. Kala. The matrix equation $AXB+CYD = E$. *Linear Algebra Appl.*, 30:141-147, 1980.

[51] Susanne M. Bane, Per Christian Hansen, and Nicholas J. Higham. A Strassen-type matrix inversion algorithm for the Connection Machine. Technical Report CNC/1993/028, Centre for Novel Computing, University of Manchester, Manchester, England, October 1993. 29 pp.

[52] C. Ballester and V. Pereyra. On the construction of discrete approximations to linear differential expressions. *Math. Comp.*, 21:297–302, 1967.

[53] Randolph E. Bank and Donald J. Rose. Marching algorithms for elliptic boundary value problems. I: The constant coefficient case. *SIAM J. Numer. Anal.*, 14(5):792-829, 1977.

[54] Yonathan Bard. *Nonlinear Parameter Estimation.* Academic Press, New York, 1974. x+341 pp. ISBN 0-12-078250-2.

[55] V. Bargmann, D. Montgomery, and J. von Neumann. Solution of linear systems of high order. Report prepared for Navy Bureau of Ordnance, 1946. Reprinted in [995, pp. 421-477].

[56] J. L. Barlow. A note on monitoring the stability of triangular decomposition of sparse matrices. *SIAM J. Sci. Statist. Comput.*, 7(1):166–168, 1986.

[57] Jesse L. Barlow. On the distribution of accumulated roundoff error in floating point arithmetic. In *Proc. 5th IEEE Symposium on Computer Arithmetic, Ann Arbor, MI*, 1981, pages 100-105.

[58] Jesse L. Barlow. *Probabilistic Error Analysis of Floating Point and CRD Arithmetics.* Ph.D. thesis, Northwestern University, Evanston, IL, USA, June 1981.

[59] Jesse L. Barlow. Error analysis and implementation aspects of deferred correction for equality constrained least squares problems. *SIAM J. Numer. Anal.,* 25(6):1340-1358, 1988.

[60] Jesse L. Barlow. On the discrete distribution of leading significant digits in finite precision arithmetic. Technical Report CS-88-35, Department of Computer Science, Pennsylvania State University, University Park, PA, USA, September 1988. 15 pp.

[61] Jesse L. Barlow. Error analysis of a pairwise summation algorithm to compute the sample variance. *Numer. Math.,* 58:583-590, 1991.

[62] Jesse L. Barlow and E. H. Bareiss. On roundoff error distributions in floating point and logarithmic arithmetic. *Computing,* 34:325–347, 1985.

[63] Jesse L. Barlow and E. H. Bareiss. Probabilistic error analysis of Gaussian elimination in floating point and logarithmic arithmetic. *Computing,* 34: 349-364, 1985.

[64] Jesse L. Barlow and Susan L. Handy. The direct solution of weighted and equality constrained least-squares problems. *SIAM J. Sci. Statist. Comput.,* 9(4):704-716, 1988.

[65] Jesse L. Barlow and Ilse C. F. Ipsen. Scaled Givens rotations for the solution of linear least squares problems on systolic arrays. *SIAM J. Sci. Statist. Comput.,* 8(5):716733, 1987.

[66] Jesse L. Barlow and Udaya B. Vemulapati. A note on deferred correction for equality constrained least squares problems. *SIAM J. Numer. Anal., 29(1):* ***249-256, 1992.***

[67] Jesse L. Barlow and Udaya B. Vemulapati. Rank detection methods for sparse matrices. *SIAM J. Matrix Anal. Appl.,* 13(4):1279-1297, 1992.

[68] S. Barnett and C. Storey. Some applications of the Lyapunov matrix equation. *J. Inst. Maths Applies,* 4:33–42, 1968.

[69] Geoff Barrett. Formal methods applied to a floating-point number system. *IEEE Trans. Soflware Engrg.,* 15(5):611-621, 1989.

[70] Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994. xiii+112 pp. ISBN 0-89871-328-5.

[71] Anders Barrlund. Perturbation bounds for the LDL^H and LU decompositions. *BIT,* 31:358–363, 1991.

[72] Anders Barrlund. How integrals can be used to derive matrix perturbation bounds. Report UMINF 92.11, Institute of Information Processing, University of Umeå, Sweden, September 1992. 8 pp.

[73] D. W. Barron and H. P. F. Swinnerton-Dyer. Solution of simultaneous linear equations using a magnetic-tape store. *Comput. J.,* 3(1):28–33, 1960.

[74] R. H. Bartels and G. W. Stewart. Algorithm 432: Solution of the matrix equation $AX + XB = C$. *Comm. ACM*, 15(9):820-826, 1972.

[75] Sven G. Bartels. Two topics in matrix analysis: Structured sensitivity for Vandermonde-like systems and a subgradient method for matrix norm estimation. M. SC. Thesis, Department of Mathematics and Computer Science, University of Dundee, Dundee, Scotland, September 1991.

[76] Sven G. Bartels and Desmond J. Higham. The structured sensitivity of Vandermonde-like systems. *Numer. Math.*, 62:17-33, 1992.

[77] Victor Barwell and Alan George. A comparison of algorithms for solving symmetric indefinite systems of linear equations. *ACM Trans. Math. Software*, 2(3):242–251, 1976.

[78] F. L. Bauer. Optimal scaling of matrices and the importance of the minimal condition. In *Proc. IFIP Congress* 1962, Cicely M. Popplewell, editor, Information Processing 62, North-Holland, Amsterdam, The Netherlands, 1963, pages 198–201.

[79] F. L. Bauer. Optimally scaled matrices. *Numer. Math.*, 5:73-87, 1963.

[80] F. L. Bauer. Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *Z. Angew. Math. Mech.*, 46(7):409-421, 1966.

[81] F. L. Bauer. Remarks on optimally scaled matrices. *Numer. Math.*, 13:1-3, 1969.

[82] F. L. Bauer. Computational graphs and rounding errors. *SIAM J. Numer. Anal.*, 11(1):87-96, 1974.

[83] F. L. Bauer and C. Reinsch. Inversion of positive definite matrices by the Gauss-Jordan method. In *Linear Algebra*, J. H. Wilkinson and C. Reinsch, editors, volume II of *Handbook for Automatic Computation*, Springer-Verlag, Berlin, 1971, pages 45-49. Contribution 1/3.

[84] F. L. Bauer, J. Steer, and C. Witzgall. Absolute and monotonic norms. *Numer. Math.*, 3:257-264, 1961.

[85] Richard M. Beam and Robert F. Warming. The asymptotic spectra of banded Toeplitz and quasi-Toeplitz matrices. *SIAM J. Sci. Comput.*, 14(4):971-1006, 1993.

[86] Albert E. Beaton, Donald B. Rubin, and John L. Barone. The acceptability of regression solutions: Another look at computational accuracy. *J. Amer. Statist. Assoc.*, 71(353):158-168, 1976.

[87] C. Gordon Bell and Allen Newell. *Computer Structures: Readings and Examples*. McGraw-Hill, New York, 1971. xix+668 pp. ISBN 07-004357-4.

[88] E. T. Bell. Review of "Contributions to the History of Determinants, 1900-1920", by Sir Thomas Muir. *Amer. Math. Monthly*, 38:161-164, 1931. Reprinted in [358].

[89] Richard Bellman. *Introduction to Matrix Analysis*. Second edition, McGraw-Hill, New York, 1970. xxiii+403 pp. First edition (1960) reprinted by Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. ISBN 0-89871-346-3.

[90] Frank Benford, The law of anomalous numbers. *Proceedings of the American Philosophical Society,* 78(4):551–572, 1938.

[91] Commandant Benoit. Note sur une méthode de resolution des équation normales provenant de l'application de la méthode des moindres carrés á un système d'équations linéaires en nombre inférieur à celui des inconnues. Application de la méthode à la résolution d'un système défini d'équations linéaires (Procédé de Commandant Cholesky). *Bulletin Géodésique (Toulouse),* 7(1):67-77, 1924. Cited in [22, Cottle's translation].

[92] N. F. Benschop and H. C. Ratz. A mean square estimate of the generated roundoff error in constant matrix iterative processes. *J. Assoc. Comput. Mach.,* 18(1):48-62, 1971.

[93] M. C. Berenbaum. Direct search methods in the optimisation of cancer chemotherapy. *Br. J. Cancer,* 61: 101–109, 1991.

[94] Abraham Berman and Robert J. Plemmons. *Nonnegative Matrices in the Mathematical Sciences.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994. xx+340 pp. Corrected republication, with supplement, of work first published in 1979 by Academic Press. ISBN O-89871-321-8.

[95] Rajendra Bhatia and Kalyan Mukherjea. Variation of the unitary part of a matrix. *SIAM J. Matrix Anal. Appl.,* 15(3):1007–1014, 1994.

[96] Rajendra Bhatia and Peter Rosenthal. How and why to solve the operator equation $AX - XB = Y$. *Bull. London Math. Soc.,* 1996. To appear.

[97] Dario Bini and Grazia Lotti. Stability of fast algorithms for matrix multiplication. *Numer. Math.,* 36:63–72, 1980.

[98] Dario Bini and Victor Y. Pan. *Polynomial and Matrix Computations. Volume 1: Fundamental Algorithms.* Birkhäuser, Boston, MA, USA, 1994. xvi+415 pp. ISBN 0-8176-3786-9.

[99] Garrett Birkhoff. Two Hadamard numbers for matrices. *Comm. ACM,* 18 (1):25-29, 1975.

[100] Garrett Birkhoff and Surender Gulati. Isotropic distributions of test matrices. *J. Appl. Math. Phys. (ZAMP),* 30:148-158, 1979.

[101] Garrett Birkhoff and Robert E. Lynch. Numerical *Solution of Elliptic Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1984. xi+319 pp. ISBN 0-89871-197-5.

[102] Garrett Birkhoff and Saunders Mac Lane. *A Survey of Modern Algebra.* Fourth edition, Macmillan, New York, 1977. xi+500 pp. ISBN 0-02-310070-2.

[103] Christian H. Bischof. Incremental condition estimation. *SIAM J. Matrix Anal. Appl.,* 11(2):312-322, 1990.

[104] Christian H. Bischof, John G. Lewis, and Daniel J. Pierce. Incremental condition estimation for sparse matrices. *SIAM J. Matrix Anal. Appl.,* 11 (4):644-659, 1990.

[105] Christian H. Bischof and Charles F. Van Loan. The WY representation for products of Householder matrices. *SIAM J. Sci. Statist. Comput.,* 8(l): s2-s13, 1987.

[106] Åke Björck. Iterative refinement of linear least squares solutions I. *BIT,* 7: 257–278, 1967.

[107] Åke Björck. Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT,* 7:1–21, 1967.

[108] Åke Björck. Iterative refinement of linear least squares solutions II. *BIT,* 8: 8-30, 1968.

[109] Åke Björck. Comment on the iterative refinement of least-squares solutions. *J. Amer. Statist. Assoc.,* 73(361):161-166, 1978.

[110] Åke Björck. Stability analysis of the method of seminormal equations for linear least squares problems. *Linear Algebra and Appl.,* 88/89:31-48, 1987.

[111] Åke Björck. Iterative refinement and reliable computing. In *Reliable Numerical Computation,* M. G. Cox and S. J. Hammarling, editors, Oxford University Press, 1990, pages 249-266.

[112] Åke Björck. Least squares methods. In *Handbook of Numerical Analysis*, P. G. Ciarlet and J. L. Lions, editors, volume I: Finite Difference Methods—Solution of Equations in $\mathbb{R}^n$, Elsevier/North-Holland, Amsterdam, The Netherlands, 1990.

[113] Åke Björck. Component-wise perturbation analysis and error bounds for linear least squares solutions. *BIT,* 31:238-244, 1991.

[114] Åke Björck. Pivoting and stability in the augmented system method. In *Numerical Analysis 1991, Proceedings of the 14th Dundee Conference,* D. F. Griffiths and G. A. Watson, editors, volume 260 of *Pitman Research Notes in Mathematics,* Longman Scientific and Technical, Essex, UK, 1992, pages 1-16.

[115] Åke Björck. Numerics of Gram-Schmidt orthogonalization. *Linear Algebra Appl.,* 197/198:297-316, 1994.

[116] Åke Björck. *Numerical Methods for Least Squares Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. To appear.

[117] Åke Björck and Tommy Elfving. Algorithms for confluent Vandermonde systems. *Numer. Math.,* 21:130-137, 1973.

[118] Åke Björck and Gene H. Golub. Iterative refinement of linear least squares solutions by Householder transformation. *BIT,* 7:322–337, 1967.

[119] Åke Björck and C. C. Paige. Loss and recapture of orthogonality in the modified Gram–Schmidt algorithm. *SIAM J. Matrix Anal. Appl.,* 13(1):176–190, 1992.

[120] Åke Björck and C. C. Paige. Solution of augmented linear systems using orthogonal factorization. *BIT,* 34:1–24, 1994.

[121] Åke Björck and Victor Pereyra. Solution of Vandermonde systems of equations. *Math. Comp.,* 24(112):893-903, 1970.

[122] P. Bjørstad, F. Manne, T. Sørevik, and M. Vajteršic. Efficient matrix multiplacation on SIMD computers. *SIAM J. Matrix Anal. Appl.,* 13(1):386-401, 1992.

[123] G. Blanch. Numerical evaluation of continued fractions. *SIAM Rev.,* 6(4): 383-421, 1964.

[124] J. H. Bleher, A. E. Roeder, and S. M. Rump. ACRITH: High-accuracy arithmetic. An advanced tool for numerical computation. In *Proceedings of the 7th Symposium on Computer Arithmetic,* Kai Hwang, editor, IEEE Computer Society Press, Silver Spring, MD, USA, 1985, pages 318-321.

[125] J. H. Bleher, S, M. Rump, U. Kulisch, M. Metzger, Ch. Ullrich, and W. Walter. A study of a FORTRAN extension for engineering/scientific computation with access to ACRITH. *Computing,* 39:93–110, 1987.

[126] B. Bliss, M.-C. Brunet, and E. Gallopoulos. Automatic program instrumentation with applications in performance and error analysis. In *Expert Systems for Scientific Computing,* E. N. Houstis, J. R. Rice, and R. Vichnevetsky, editors, North-Holland, Amsterdam, The Netherlands, 1992, pages 235–260.

[127] Richard M. Bloch. Mark I calculator. In [507], pages 23-30.

[128] James L. Blue. A portable Fortran program to find the Euclidean norm of a vector. *ACM Trans. Math. Software,* 4(1):15–23, 1978.

[129] E. Bodewig. Review of “Rounding-Off Errors in Matrix Processes” by A. M. Turing. *Math. Rev.,* 10:405, 1949.

[130] Gerd Bohlender. Floating-point computation of functions with maximum accuracy. *IEEE Trans. Comput.,* C-26(7):621–632, 1977.

[131] Z. Bohte. Bounds for rounding errors in the Gaussian elimination for band systems. *J. Inst. Maths Applics,* 16:133–142, 1975.

[132] Daniel Boley, Gene H. Golub, Samy Makar, Nirmal Saxena, and Edward J. McCluskey. Floating point fault tolerance with backward error assertions. *IEEE Trans. Comput.,* 44(2):302-311, 1994.

[133] Jo A. M. Bollen. Numerical stability of descent methods for solving linear equations. *Numer. Math.,* 43:361–377, 1984.

[134] S. Bondeli and W. Gander. Cyclic reduction for special tridiagonal systems. *SIAM J. Matrix Anal. Appl.,* 15(1):321-330, 1994.

[135] T. Boros, T. Kailath, and V. Olshevsky. Fast algorithms for solving Vandermonde and Chebyshev–Vandermonde systems. Manuscript, 1994. 22 pp.

[136] J. M. Borwein, P. B. Borwein, and D. H. Bailey. Ramanujan, modular equations, and approximations to Pi or how to compute one billion digits of Pi. *Amer. Math. Monthly,* 96(3):201-219, 1989.

[137] B. V. Bowden. The organization of a typical machine. In *Faster than Thought: A Symposium on Digital Computing Machines,* B. V. Bowden, editor, Pitman, London, 1953, pages 67–77.

[138] H. J. Bowdler, R. S. Martin, G. Peters, and J. H. Wilkinson. Solution of real and complex systems of linear equations. *Numer. Math.*, 8:217–234, 1966. Also in [1102, pp. 93-110], Contribution 1/7.

[139] David W. Boyd. The power method for ℓ^p norms. *Linear Algebra Appl.*, 9: 95-101, 1974.

[140] Carl B. Boyer. A *History of Mathematics*. Wiley, New York, 1968. xv+717 pp. Reprinted by Princeton University Press, Princeton, NJ, USA, 1985. ISBN 0-691-02391-3.

[141] Jeff Boyle. An application of Fourier series to the most significant digit problem. *Amer. Math. Monthly*, 101(11):879-886, 1994.

[142] R. P. Brent. Algorithms for matrix multiplication. Technical Report CS 157, Department of Computer Science, Stanford University, Stanford, CA, USA, March 1970. ii+52 pp.

[143] R. P. Brent. Error analysis of algorithms for matrix multiplication and triangular decomposition using Winograd's identity. *Numer. Math.*, 16:145-156, 1970.

[144] Richard P. Brent. On the precision attainable with various floating-point number systems. *IEEE Trans. Comput.*, C-22(6):601-607, 1973.

[145] Richard P. Brent. A Fortran multiple-precision arithmetic package. *ACM Trans. Math. Software*, 4(1):57-70, 1978.

[146] Richard P. Brent. ALGORITHM 524 MP, a Fortran multiple-precision arithmetic package. *ACM nuns. Math. Software*, 4(1):71–81, 1978.

[147] Richard P. Brent, Judith A. Hooper, and J. Michael Yohe. An AUGMENT interface for Brent's multiple precision arithmetic package. *ACM Trans. Math. Software*, 6(2):146-149, 1980.

[148] William L. Briggs and Van Emden Henson. *The DFT: An Owner's Manual for the Discrete Fourier Transform*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xv+434 pp. ISBN 0-89871-342-0.

[149] J. L. Britton, editor. *Collected Works of A. M. Turing: Pure Mathematics*. North-Holland, Amsterdam, The Netherlands, 1992. xxii+ 287 pp. ISBN 0-44488059-3.

[150] W. S. Brown. A simple but realistic model of floating-point arithmetic. *ACM Trans. Math. Software*, 7(4):445–480, 1981.

[151] Shirley V. Browne, Jack J. Dongarra, Stan C. Green, Keith Moore, Thomas H. Rowan, and Reed C. Wade. Netlib services and resources. Report ORNL/TM-12680, Oak Ridge National Laboratory, Oak Ridge, TN, USA, April 1994.42 pp.

[152] Marie-Christine Brunet. *Contribution* à *la Fiabilité de Logiciels Numériques et à L'analyse de Leur Comportement: Une Approche Statistique*. Ph.D. thesis, Université de Paris IX Dauphine, U.E.R. Mathématiques de la Décision, January 1989. viii+214 pp.

[153] Marie-Christine Brunet and Françoise Chatelin. CESTAC, a tool for a stochastic round-off error analysis in scientific computing. In Numerical *Mathematics and Applications,* R. Vichnevetsky and J. Vignes, editors, Elsevier Science Publishers B.V. (North-Holland), Amsterdam, The Netherlands, 1986, pages 11–20.

[154] James L, Buchanan and Peter R. Turner. Numerical *Methods and Analysis.* McGraw-Hill, New York, 1992. xv+751 pp. ISBN 0-07-008717-2, 0-07-112922-7 (international paperback edition).

[155] W. Buchholz. Fingers or fists? (The choice of decimal or binary representation). *Comm. ACM,* 2(12):3–11, 1959.

[156] W. Buchholz. Origin of the word *byte. Ann. Hist. Comput.,* 3(1):72, 1981.

[157] B. Bukhberger and G. A. Emel'yanenko. Methods of inverting tridiagonal matrices. *U.S. S. R. Computational Math. Math. Phys.,* 13:10–20, 1973.

[158] James R. Bunch. Analysis of the diagonal pivoting method. *SIAM J. Numer. Anal.,* 8(4):656-680, 1971.

[159] James R. Bunch. Equilibration of symmetric matrices in the max-norm. *J. Assoc. Comput. Mach.,* 18(4):566-572, 1971.

[160] James R. Bunch. Partial pivoting strategies for symmetric matrices. *SIAM J. Numer. Anal.,* 11(3):521-528, 1974.

[161] James R. Bunch. A note on the stable decomposition of skew-symmetric matrices. *Math. Comp.,* 38(158):475–479, 1982.

[162] James R. Bunch. The weak and strong stability of algorithms in numerical linear algebra. *Linear Algebra Appl.,* 88/89:49-66, 1987.

[163] James R. Bunch, James W. Demmel, and Charles F. Van Loan. The strong stability of algorithms for solving symmetric linear systems. *SIAM J. Matrix Anal. Appl.,* 10(4):494-499, 1989.

[164] James R. Bunch and Linda Kaufman. Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.,* 31(137):163–179, 1977.

[165] James R. Bunch, Linda Kaufman, and Beresford N. Parlett. Decomposition of a symmetric matrix. Numer. *Math.,* 27:95–109, 1976.

[166] James R. Bunch and Beresford N. Parlett. Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.,* 8(4): 639-655, 1971.

[167] P. Businger and G. H. Golub. Linear least squares solutions by Householder transformations. *Numer. Math.,* 7:269-276, 1965. Also in [1102, pp. 111-118], Contribution I/8.

[168] P. A. Businger. Matrices which can be optimally scaled. *Numer. Math.,* 12: 346-348, 1968.

[169] P. A. Businger. Monitoring the numerical stability of Gaussian elimination. *Numer. Math.,* 16:36@361, 1971.

[170] J. C. Butcher. *The Numerical Analysts of Ordinary Differential Equations: Runge-Kutta and General Linear Methods.* John Wiley, Chichester, UK, 1987. xv+512 pp. ISBN 0-471-91046-5.

[171] B. L. Buzbee, G. H. Golub, and C. W. Nielson. On direct methods for solving Poisson's equations. *SIAM J. Numer. Anal.,* 7(4):627-656, 1970.

[172] Ralph Byers. A LINPACK-style condition estimator for the equation $AX - XB^T = C$. *IEEE Trans. Automat. Control,* AC-29(10):926-928, 1984.

[173] Ralph Byers. Numerical condition of the algebraic Riccati equation. In *Linear Algebra and its Role in Systems Theory,* B. N. Datta, editor, volume 47 of *Contemporary Math.,* American Mathematical Society, Providence, RI, USA, 1985, pages 3549.

[174] Ralph Byers. Solving the algebraic Riccati equation with the matrix sign function. *Linear Algebra Appl.,* 85:267-279, 1987.

[175] Ralph Byers. A bisection method for measuring the distance of a stable matrix to the unstable matrices. *SIAM J. Sci. Statist. Comput.,* 9:875-881, 1988.

[176] Ralph Byers and Stephen Nash. On the singular "vectors" of the Lyapunov operator. *SIAM J. Alg. Discrete Methods,* 8(1):59-66, 1987.

[177] John Caffney. Another test matrix for determinants and inverses. *Comm. ACM,* 6(6):310, 1963.

[178] D. Calvetti and L. Reichel. A Chebychev-Vandermonde solver. *Linear Algebra Appl.,* 172:219-229, 1992.

[179] D. Calvetti and L. Reichel. Fast inversion of Vandermonde-like matrices involving orthogonal polynomials. *BIT,* 33:473–484, 1993.

[180] S. L. Campbell and C. D. Meyer, Jr. *Generalized Inverses of Linear Transformations.* Pitman, London, 1979. xi+272 pp. Reprinted by Dover, New York, 1991. ISBN 0-486-66693-X.

[181] Martin Campbell-Kelly. Programming the Pilot ACE: Early programming activity at the National Physical Laboratory. *Ann. Hist. Comput.,* 3(2): 133–162, 1981.

[182] Martin Campbell-Kelly. Review of "Alan Turing: The Enigma", by Andrew Hodges. *Ann. Hist. Comput.,* 6(2):176–178, 1984.

[183] Claudio Canuto, M. Yousuff Hussaini, Alfio Quarteroni, and Thomas A. Zang. *Spectral Methods in Fluid Dynamics.* Springer-Verlag, Berlin, 1988. xv+567 pp. ISBN 3-540-52205-0.

[184] Wei-Lu Cao and William J. Stewart. A note on inverses of Hessenberg-like matrices. *Linear Algebra Appl.,* 76:233–240, 1986.

[185] Ole Caprani. Implementation of a low round-off summation method. *BIT,* 11:271–275, 1971.

[186] B. E. Carpenter and R. W. Doran, editors. *A. M. Turing's ACE Report of 1946 and Other Papers,* volume 10 of *Charles Babbage Institute Reprint Series for the History of Computing.* MIT Press, Cambridge, MA, USA, 1986. vii+141 pp. ISBN 0-262-031140.

[187] John W. Carr III. Error analysis in floating point arithmetic. Comm. *ACM*, 2(5):10-15, 1959.

[188] Russell Carter. Y-MPfloating point and Cholesky factorization. *Internat. J. High Speed Computing*, 3(3/4):215-222, 1991.

[189] A. Catchy. *Exercices d'Analyse et de Phys. Math.*, volume 2. Paris, 1841. Cited in [896].

[190] Françoise Chaitin-Chatelin and Valérie Frayssé. *Lectures on Finite Precision Computations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996. To appear.

[191] Raymond H. Chan, James G. Nagy, and Robert J. Plemmons. Circulant preconditioned Toeplitz least squares iterations. *SIAM J. Matrix Anal. Appl.*, 15(1):80-97, 1994.

[192] T. F. Chan and P. C. Hansen. Some applications of the rank revealing QR factorization. *SIAM J. Sci. Statist. Comput.*, 13(3):727-741, 1992.

[193] Tony F. Chan and David E. Foulser. Effectively well-conditioned linear systems. *SIAM J. Sci. Statist. Cornput.*, 9(6):963–969, 1988.

[194] Tony F. Chan, Gene H. Golub, and Randall J. LeVeque. Algorithms for computing the sample variance: Analysis and recommendations. *Amer. Statist.*, 37(3):242-247, 1983.

[195] Tony F. Chan and John Gregg Lewis. Computing standard derivations: Accuracy. *Comm. ACM*, 22(9):526–531, 1979.

[196] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. Backward errors for eigenvalue and singular value decompositions. *Numer. Math.*, 68:215–223, 1994.

[197] Shivkumar Chandrasekaran and Ilse C. F. Ipsen. On the sensitivity of solution components in linear systems of equations. *SIAM J. Matrix Anal. Appl.*, 16(1):93-112, 1995.

[198] Xiao-Wen Chang and C. C. Paige. New perturbation bounds for the Cholesky factorization. Manuscript, February 1995. 13 pp.

[199] Bruce W. Char, Keith O. Geddes, Gaston H. Gonnet, Benton L. Leong, Michael B. Monagan, and Stephen M. Watt. *Maple V Library Reference Manual*. Springer-Verlag, Berlin, 1991. xxv+698 pp. ISBN 3-540-97592-6.

[200] Bruce A. Chartres and James C. Geuder. Computable error bounds for direct solution of linear equations. *J. Assoc. Comput. Mach.*, 14(1):63–71, 1967.

[201] Françoise Chatelin. *Eigenvalues of Matrices*. Wiley, Chichester, UK, 1993. xviii+382 pp. ISBN 0-471-93538-7.

[202] Françoise Chatelin and Marie-Christine Brunet. A probabilistic round-off error propagation model. Application to the eigenvalue problem. In *Reliable Numerical Computation*, M. G. Cox and S. J. Hammarling, editors, Oxford University Press, Oxford, UK, 1990, pages 139-160.

[203] Françoise Chatelin and Valérie Frayssé. Elements of a condition theory for the computational analysis of algorithms. In *Iterative Methods in Linear Algebra*, R. Beauwens and P. de Green, editors, Elsevier (North-Holland), Amsterdam, The Netherlands, 1992, pages 15–25.

[204] Denise Chen and Cleve Moler. *Symbolic Math Toolbox: User's Guide.* The MathWorks, Inc., Natick, MA, USA, 1993.

[205] Jaeyoung Choi, Jack J. Dongarra, Susan Ostrouchov, Antoine P. Petitet, David W. Walker, and R. Clint Whaley. The design and implementation of the ScaLAPACK LU, QR and Cholesky factorization routines. Report 0RNL/TM-12470, Oak Ridge National Laboratory, Oak Ridge, TN, USA, September 1994. 26 pp. LAPACK Working Note 80.

[206] Jaeyoung Choi, Jack J. Dongarra, Roldan Pozo, and David W. Walker. ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers. Technical Report CS-92-181, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, November 1992.8 pp. LAPACK Working Note 55.

[207] Man-Duen Choi. Tricks or treats with the Hilbert matrix. *Amer. Math.* Monthly 90:301-312, 1983.

[208] Søren Christiansen and Per Christian Hansen. The effective condition number applied to error analysis of certain boundary collocation methods. *J. Comp. Appl. Math.,* 54:15-36, 1994.

[209] Eleanor Chu and Alan George. A note on estimating the error in Gaussian elimination without pivoting. *ACM SIGNUM Newsletter,* 20(2):2–7, 1985.

[210] King-wah Eric Chu. The solution of the matrix equations $AXB - CXD = E$ and $(YA - DZ, YC - BZ) = (E, F)$. *Linear Algebra Appl.,* 93:93–105, 1987.

[211] Barry A. Cipra. Computer-drawn pictures stalk the wild trajectory. Science, 241:1162–1163, 1988.

[212] *C.* W. Clenshaw. A note on the summation of Chebyshev series. *M.T.A.C.,* 9(51):118-120, 1955.

[213] C. W. Clenshaw and F. W. J. Olver. Beyond floating point. *J. Assoc. Comput. Mach.,* 31(2):319-328, 1984.

[214] C. W. Clenshaw, F. W. J. Olver, and P. R. Turner. Level-index arithmetic: An introductory survey. In Numerical *Analysis and Parallel Processing, Lancaster 1987,* Peter R. Turner, editor, volume 1397 of *Lecture Notes in Mathematics,* Springer-Verlag, Berlin, 1989, pages 95–168.

[215] A. K. Cline. An elimination method for the solution of linear least squares problems. *SIAM J. Numer. Anal.,* 10(2):283-289, 1973.

[216] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM J. Numer. Anal., 16(2):368-375, 1979.*

[217] A. K. Cline and R. K. Rew. A set of counter-examples to three condition number estimators. *SIAM J. Sci. Statist. Comput.,* 4(4):602-611, 1983.

[218] Alan K. Cline, Andrew R. Corm, and Charles F. Van Loan. Generalizing the LINPACK condition estimator. In *Numerical Analysis, Mexico* 1981, J. P. Hennart, editor, volume 909 of *Lecture Notes in Mathematics,* Springer-Verlag, Berlin, 1982, pages 73-83.

[219] R. E. Cline and R. J. Plemmons. l_2-solutions to underdetermined linear systems. *SIAM Rev.,* 18(1):92–106, 1976.

[220] William D. Clinger. How to read floating point numbers accurately. *SIGPLAN Notices,* 25(6):92-101, 1990.

[221] W. J. Cody. Implementation and testing of function software. In *Problems and Methodologies in Mathematical Software Production,* Paul C. Messina and Almerico Murli, editors, volume 142 of *Lecture Notes in Computer Science,* Springer-Verlag, Berlin, 1982, pages 24–47.

[222] W. J. Cody. ALGORITHM 665 MACHAR: A subroutine to dynamically determine machine parameters. *ACM Trans. Math. Software,* 14(4):303-311, 1988.

[223] W. J. Cody. Floating-point standards-Theory and practice. In *Reliability in Computing: The Role of Interval Methods in Scientific Computing,* Ramon E. Moore, editor, Academic Press, Boston, MA, USA, 1988, pages 99-107.

[224] W. J. Cody. Algorithm 714. CELEFUNT: A portable test package for complex elementary functions. *ACM Trans. Math. Software,* 19(1):121, 1993.

[225] W. J. Cody, J. T. Coonen, D. M. Gay, K. Hanson, D. Hough, W. Kahan, R. Karpinski, J. Palmer, F. N. Ris, and D. Stevenson. A proposed radix- and word-length-independent standard for floating-point arithmetic. *IEEE Micro,* 4(4):86-100, 1984.

[226] W. J. Cody and Jerome T. Coonen. ALGORITHM 722: Functions to support the IEEE standard for binary floating-point arithmetic. *ACM Trans. Math. Software,* 19(4):443-451, 1993.

[227] William J. Cody, Jr. Static and dynamic numerical characteristics of floating-point arithmetic. *IEEE Trans. Comput.,* C-22(6):598-601, 1973.

[228] William J. Cody, Jr. and William Waite. *Software Manual for the Elementary Functions.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1980. x+269 pp. ISBN 0-13-822064-6.

[229] A. M. Cohen. A note on pivot size in Gaussian elimination. *Linear Algebra Appl.,* 8:361-368, 1974.

[230] A. M. Cohen. The inverse of a Pascal matrix. *Mathematical Gazette,* 59 (408):111-112, 1975.

[231] Marty S. Cohen, T. E. Hull, and V. Carl Hamacher. CADAC: A controlled-precision decimal arithmetic unit. *IEEE Trans. Comput.,* C-32(4):370-377, 1983.

[232] Thomas F. Coleman and Charles F. Van Loan. *Handbook for Matrix Computations.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1988. vii+264 pp. ISBN 0-89871-227-0.

[233] Apple Computer. *Apple Numerics Manual.* Second edition, Addison-Wesley, Reading, MA, USA, 1988.

[234] Apple Computer. *Inside Macintosh: PowerPC Numerics.* Addison-Wesley, Reading, MA, USA, 1994. ISBN 0-201-40728-0.

[235] P. Concus, G. H. Golub, and G. Meurant. Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Statist. Comput.*, 6(1):220-252, 1985.

[236] Andrew R. Corm, Nicholas I. M. Gould, and Philippe L. Toint. *LANCELOT: A Fortran Package for Large-Scale Nonlinear Optimization (Release A).* Springer-Verlag, Berlin, 1992. xviii+330 pp. ISBN 0-387-55470-X.

[237] Samuel D. Conte and Carl de Boor. *Elementary Numerical Analysis: An Algorithmic Approach.* Third edition, McGraw-Hill, Tokyo, 1980. xii+432 pp. ISBN 0-07-066228-2.

[238] James W. Cooley. How the FFT gained acceptance. In *A History of Scientific Computing,* Stephen G. Nash, editor, Addison-Wesley, Reading, MA, USA, 1990, pages 133-140.

[239] James W. Cooley. Lanczos and the FFT: A discovery before its time. In *Proceedings of the Cornelius Lanczos International Centenary Conference,* J. David Brown, Moody T. Chu, Donald C. Ellison, and Robert J. Plemmons, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994, pages 3-9.

[240] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comp.,* 19(90):297-301, 1965.

[241] James W. Cooley and John W. Tukey. On the origin and publication of the FFT paper. *Current Contents,* (51-52):8-9, 1993.

[242] Brian A. Coomes, Huseyin Koçak, and Kenneth J. Palmer. Rigorous computational shadowing of orbits of ordinary differential equations. *Numer. Math.,* 69(4):401-421, 1995.

[243] Jerome T, Coonen. Underflow and the denormalized numbers. *Computer,* 14:75-87, 1981.

[244] J. E. Cope and B. W. Rust. Bounds on solutions of linear systems with inaccurate data. *SIAM J. Numer. Anal.,* 16(6):950-963, 1979.

[245] Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing,* ACM Press, New York, 1987, pages 1-6.

[246] Rob Corless. Continued fractions and chaos. *Amer. Math. Monthly,* 99(3): 203-215, 1992.

[247] Robert M. Corless. Defect-controlled numerical methods and shadowing for chaotic differential equations. *Physics D,* 60:323–334, 1992.

[248] Richard W. Cottle. Manifestations of the Schur complement. *Linear Algebra Appi.,* 8:189-211, 1974.

[249] M. G. Cox. The numerical evaluation of *B*-splines. *J. Inst. Maths Applies,* 10:134–149, 1972.

[250] M. G. Cox. An algorithm for spline interpolation. *J. Inst. Maths Applies,* 15:95–108, 1975.

[251] M. G. Cox. The numerical evaluation of a spline from its B-spline representation. *J. Inst. Maths Applics,* 21:135–143, 1978.

[252] M. G. Cox and S. J. Hammarling, editors. *Reliable Numerical Computation.* Oxford University Press, Oxford, UK, 1990. xvi+339 pp. ISBN 0-19-853564-3.

[253] M. G. Cox and P. M. Harris. Overcoming an instability arising in a spline approximation algorithm by using an alternative form of a simple rational function. *IMA Bulletin,* 25(9):228-232, 1989.

[254] Fred D. Crary. A versatile precompiled for nonstandard arithmetics. *ACM Trans. Math. Software,* 5(2):204-217, 1979.

[255] Prescott D. Crout. A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients. *Trans. Amer. Inst. Elec. Engrg.,* 60:1235-1241, 1941.

[256] C. W. Cryer. Pivot size in Gaussian elimination. *Numer. Math.,* 12:335-345, 1968.

[257] Colin W. Cryer. The *LU*-factorization of totally positive matrices. *Linear Algebra Appl.,* 7:83-92, 1973.

[258] Colin W. Cryer. Some properties of totally positive matrices. *Linear Algebra Appl.,* 15:1-25, 1976.

[259] A. R. Curtis and J. K. Reid. On the automatic scaling of matrices for Gaussian elimination. *J. Inst. Maths Applics,* 10:118–124, 1972.

[260] George Cybenko and Charles F. Van Loan. Computing the minimum eigenvalue of a symmetric positive definite Toeplitz matrix. *SIAM J. Sci. Statist. Comput.,* 7(1):123-131, 1986.

[261] G. Dahlquist. On matrix majorants and minorants, with applications to differential equations. *Linear Algebra Appl.,* 52/53:199-216, 1983.

[262] Germund Dahlquist and Åke Björck. *Numerical Methods.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1974. xviii+573 pp. Translated by Ned Anderson. ISBN 0-13-627315-7.

[263] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.,* 30(136):772–795, 1976.

[264] B. Danloy. On the choice of signs for Householder's matrices. *J. Comp. Appl. Math.,* 2(1):67-69, 1976.

[265] Karabi Datta. The matrix equation $XA - BX = R$ and its applications. *Linear Algebra Appl.,* 109:91-105, 1988.

[266] Philip J. Davis. *Circulant Matrices.* Wiley, New York, 1979. xv+250 pp. ISBN 0-471-05771-1.

[267] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration.* Second edition, Academic Press, London, 1984. xiv+612 pp. ISBN 0-12-206360-0.

[268] Achiya Dax. Partial pivoting strategies for symmetric Gaussian elimination. *Math. Programming,* 22:288-303, 1982.

[269] Achiya Dax. The convergence of linear stationary iterative processes for solving singular unstructured systems of linear equations. *SIAM Rev.,* 32(4): 611-635, 1990.

[270] Achiya Dax and S. Kaniel. Pivoting techniques for symmetric Gaussian elimination. *Numer. Math.,* 28:221-241, 1977.

[271] Jane M. Day and Brian Peterson. Growth in Gaussian elimination. *Amer. Math. Monthly,* 95(6):489–513, 1988.

[272] Carl de Boor. On calculating with *B*-splines. *J. Approx. Theory,* 6:5042, 1972.

[273] Carl de Boor and Allan Pinkus. Backward error analysis for totally positive linear systems. *Numer. Math.,* 27:485-490, 1977.

[274] Lieuwe Sytse de Jong. Towards a formal definition of numerical stability. *Numer. Math.,* 28:211-219, 1977.

[275] T. J. Dekker. A floating-point technique for extending the available precision. *Numer. Math,,* 18:224-242, 1971.

[276] T. J. Dekker and W. Hoffmann. Rehabilitation of the Gauss-Jordan algo rithm. *Numer. Math.,* 54:591–599, 1989.

[277] Cédric J. Demeure. Fast *QR* factorization of Vandermonde matrices. *Linear Algebra and Appl.,* 122/3/4:165-194, 1989.

[278] Cédric J. Demeure. *QR* factorization of confluent Vandermonde matrices. *IEEE trans. Acoust., Speech, Signal Processing,* 38(10):1799-1802, 1990.

[279] James W. Demmel. The condition number of equivalence transformations that block diagonalize matrix pencils. *SIAM J. Numer. Anal.,* 20(3):599-610, 1983.

[280] James W. Demmel. Underflow and the reliability of numerical software. *SIAM J. Sci. Statist. Comput.,* 5(4):887-919, 1984.

[281] James W. Demmel. On condition numbers and the distance to the nearest ill-posed problem. *Numer. Math.,* 51:251–289, 1987.

[282] James W. Demmel. On error analysis in arithmetic with varying relative precision. In *Proceedings of the Eighth Symposium on Computer Arithmetic, Como, Italy,* Mary Jane Irwin and Renato Stefanelli, editors, IEEE Computer Society, Washington, DC, 1987, pages 148-152.

[283] James W. Demmel. On floating point errors in Cholesky. Technical Report CS-89-87, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, October 1989.6 pp. LAPACK Working Note 14.

[284] James W. Demmel. On the odor of IEEE arithmetic. *NA Digest,* Volume 91, Issue 39, 1991. (Response to a message “IEEE Arithmetic Stinks” in Volume 91, Issue 33). Electronic mail magazine: `na.helptia-net.ornl.gov.`

[285] James W. Demmel. The componentwise distance to the nearest singular matrix. *SIAM J. Matrix Anal. Appl.,* 13(1):10-19, 1992.

[286] James W. Demmel. Open problems in numerical linear algebra. IMA Preprint Series #961, Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN, USA, April 1992. 21 pp. LAPACK Working Note 47.

[287] James W. Demmel. A specification for floating point parallel prefix. Technical Report CS-92-167, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, July 1992. 8 pp. LAPACK Working Note 49.

[288] James W. Demmel. Trading off parallelism and numerical stability. In *Linear Algebra for Larqe Scale and Real-Time Applications,* Marc S, Moonen, Gene H. Golub, and Bart L. De Moor, editors, volume 232 of *NATO ASI Series E,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pages 49-68.

[289] James W. Demmel, Inderjit Dhillon, and Huan Ren. On the correctness of parallel bisection in floating point. Technical Report CS-94-228, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, March 1994. 38 pp. LAPACK Working Note 70.

[290] James W. Demmel, J. J. Dongarra, and W. Kahan. On designing portable high performance numerical libraries. In Numerical *Analysis 1991, Proceedings of the 14th Dundee Conference,* D. F. Griffiths and G. A. Watson, editors, volume 260 of *Pitman Research Notes in Mathematics,* Longman Scientific and Technical, Essex, UK, 1992, pages 69-84.

[291] James W. Demmel and Nicholas J. Higham. Stability of block algorithms with fast level-3 BLAS. *ACM Trans. Math. Software,* 18(3):274–291, 1992.

[292] James W. Demmel and Nicholas J. Higham. Improved error bounds for underdetermined system solvers. *SIAM J. Matrix Anal. Appl.,* 14(1):1-14, 1993.

[293] James W. Demmel, Nicholas J. Higham, and Robert S. Schreiber. Stability of block *LU* factorization. Numerical *Linear Algebra with Applications,* 2(2): 173-190, 1995.

[294] James W. Demmel and Bo Kågström. Computing stable eigendecompositions of matrix pencils. *Linear Algebra Appl.,* 88/89:139-186, 1987.

[295] James W. Demmel and Bo Kågström. Accurate solutions of ill-posed problems in control theory. *SIAM J. Matrix Anal. Appl.,* 9(1):126-145, 1988.

[296] James W. Demmel and W. Kahan. Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Statist. Comput.,* 11(5):873–912, 1990.

[297] James W. Demmel and F. Krückeberg. An interval algorithm for solving systems of linear equations to prespecified accuracy. *Computing,* 34:117–129, 1985.

[298] James W. Demmel and Xiaoye Li. Faster numerical algorithms via exception handling. *IEEE Trans. Comput.,* 43(8):983-992, 1994.

[299] James W. Demmel and A. McKenney. A test matrix generation suite. Preprint MCS-P69-0389, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA, March 1989.16 pp. LAPACK Working Note 9.

[300] J. E. Dennis, Jr. and Robert B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1983. xiii+378 pp. ISBN 0-13-627216-9.

[301] J. E. Dennis, Jr. and Virginia Torczon. Direct search methods on parallel machines. *SIAM J. Optim.,* 1(4):448-474, 1991.

[302] J. E. Dennis, Jr. and Homer F. Walker. Inaccuracy in quasi-Newton methods: Local improvement theorems. *Math. Ping. Study,* 22:70-85, 1984.

[303] John E. Dennis, Jr. and Daniel J. Woods. Optimization on microcomputers: The Nelder-Mead simplex algorithm. In New *Computing Environments: Microcomputers in Large-Scale Computing,* Arthur Wouk, editor, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1987, pages 116-122.

[304] J. Descloux. Note on the round-off errors in iterative processes. *Math. Comp.,* 17:18-27, 1963.

[305] Harold G. Diamond. Stability of rounded off inverses under iteration. *Math. Comp.,* 32(141):227-232, 1978.

[306] John D. Dixon. Estimating extremal eigenvalues and condition numbers of matrices. *SIAM J. Numer. Anal.,* 20(4):812–814, 1983.

[307] J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart. *LINPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. ISBN 0-89871-172-X.

[308] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. J. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Thins. Math. Software, 16: 1–17, 1990.*

[309] J. J. Dongarra, J. J. Du Croz, I. S. Duff, and S. J. Hammarling. Algorithm 679. A set of level 3 basic linear algebra subprograms: Model implementation and test programs. *ACM Trans. Math. Software,* 16:18–28, 1990.

[310] J. J. Dongarra, F. G. Gustavson, and A. Karp. Implementing linear algebra algorithms for dense matrices on a vector pipeline machine. *SIAM Rev.,* 26 (1):91-112, 1984.

[311] Jack Dongarra, Roldan Pozo, and David W. Walker. LAPACK++ V1.0: High performance linear algebra users' guide. Technical Report CS-95-290, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, May 1995. 31 pp. LAPACK Working Note 98.

[312] Jack J. Dongarra. Performance of various computers using standard linear equations software. Technical Report CS-89-85, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, February 1995. 34 PP.

[313] Jack J. Dongarra, Jeremy J. Du Croz, Sven J. Hammarling, and Richard J. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Trans. Math. Software,* 14(1):1-17, 1988.

[314] Jack J. Dongarra, Jeremy J. Du Croz, Sven J. Hammarling, and Richard J. Hanson. Algorithm 656. An extended set of Fortran basic linear algebra subprograms: Model implementation and test programs. *ACM Trans. Math. Software,* 14(1):18-32, 1988.

[315] Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk A. van der Vorst. Solving *Linear Systems on Vector and Shared Memory Computers.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991. x+256 pp. ISBN 0-89871-270-X.

[316] Jack J. Dongarra and Eric Grosse. Distribution of mathematical software via electronic mail. *Comm. ACM,* 30(5):403–407, 1987.

[317] Craig C. Douglas, Michael Heroux, Gordon Slishman, and Roger M. Smith. GEMMW: A portable level 3 BLAS Winograd variant of Strassen's matrix–matrix multiply algorithm. *J. Comput. Phys.,* 110:1–10, 1994.

[318] Craig C. Douglas and Gordon Slishman. Variants of matrix-matrix multiplication for Fortran-90. *ACM SIGNUM Newsletter,* 29:4–6, 1994.

[319] Jim Douglas, Jr. Round-off error in the numerical solution of the heat equation. *J. Assoc. Comput. Mach.,* 6:48–58, 1959.

[320] Thomas C. Doyle. Inversion of symmetric coefficient matrix of positive-definite quadratic form. *M.T.A.C.,* 11:55-58, 1957.

[321] Zlatko Drma, Matjaž Omladič, and Krešimir Veselić. On the perturbation of the Cholesky factorization. *SIAM J. Matrix Anal. Appl.,* 15(4):1319–1332, 1994.

[322] Jeremy J. Du Croz and Nicholas J. Higham. Stability of methods for matrix inversion. *IMA J. Numer. Anal.,* 12:1–19, 1992.

[323] Augustin A. Dubrulle. A class of numerical methods for the computation of Pythagorean sums. *IBM J. Res. Develop.,* 27(6):582-589, 1983.

[324] Augustin A. Dubrulle and Gene H. Golub. A multishift QR iteration without computation of the shifts. *Numerical Algorithms, 7: 173–181, 1994.*

[325] I. S. Duff, A.M. Erisman, and J. K. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, 1986. xiii+341 pp. ISBN 0-19-853408-6.

[326] I. S. Duff, N. I. M. Gould, J. K. Reid, J. A. Scott, and K. Turner. The factorization of sparse symmetric indefinite matrices. *IMA J. Numer. Anal.,* 11:181–204, 1991.

[327] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Software,* 15(1):1-14, 1989.

[328] Iain S. Duff, Roger G. Grimes, and John G. Lewis. Users' guide for the Harwell-Boeing sparse matrix collection (release 1). Report RAL-92-086, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon, UK, December 1992. 84 pp.

[329] Iain S. Duff and John K. Reid. MA27—A set of Fortran subroutines for solving sparse symmetric sets of linear equations. Technical Report AERE R10533, AERE Harwell Laboratory, 1982.

[330] Iain S. Duff and John K. Reid. MA47, aFortran code for direct solution of indefinite sparse symmetric linear systems. Report RAL-95-001, Atlas Centre, Rutherford Appleton Laboratory, Didcot, Oxon, UK, January 1995. 63 pp.

[331] Iain S. Duff, John K. Reid, Neils Munskgaard, and Hans B. Nielsen. Direct solution of sets of linear equations whose matrix is sparse, symmetric and indefinite. *J. Inst. Maths Applics,* 23:235–250, 1979.

[332] William Dunham. *Journey Through Genius: The Great Theorems of Mathematics.* Penguin, New York, 1990. xiii+300 pp. ISBN 0-14-014739-X.

[333] Paul S. Dwyer. *Linear Computations.* Wiley, New York, 1951. xi+344 pp.

[334] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. Psychometrika, 1(3):211–218, 1936.

[335] Alan Edelman. Eigenvalues and condition numbers of random matrices. *SIAM J. Matrix Anal. Appl.,* 9(4):543-560, 1988.

[336] Alan Edelman. The distribution and moments of the smallest eigenvalue of a random matrix of Wishart type. *Linear Algebra Appl.,* 159:55–80, 1991.

[337] Alan Edelman. The first annual large dense linear system survey. *ACM SIGNUM Newsletter,* 26:6-12, October 1991.

[338] Alan Edelman. The complete pivoting conjecture for Gaussian elimination is false. *The Mathematical Journal,* 2:58–61, 1992.

[339] Alan Edelman. On the distribution of a scaled condition number. *Math. Comp.,* 58(197):185-190, 1992.

[340] Alan Edelman. Eigenvalue roulette and random test matrices. In *Linear Algebra for Large Scale and Real-Time Applications,* Marc S. Moonen, Gene H. Golub, and Bart L. De Moor, editors, volume 232 of *NATO ASI Series E,* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993, pages 365-368.

[341] Alan Edelman. Large dense numerical linear algebra in 1993: The parallel computing influence. *Int. J. Supecomputer Appl.,* 7(2): 113–128, 1993.

[342] Alan Edelman. Scalable dense numerical linear algebra in 1994: The multicomputer influence. In *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra,* John G. Lewis, editor, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994, pages 344-348.

[343] Alan Edelman. When is $x * (1/x) \neq 1$? Manuscript, 1994.

[344] Alan Edelman, Eric Kostlan, and Michael Shub. How many eigenvalues of a random matrix are real? *J. Amer. Math. Soc.,* 7(1):247–267, 1994.

[345] Alan Edelman and Walter Mascarenhas. On the complete pivoting conjecture for a Hadamard matrix of order 12. *Linear and Multilineal Algebra,* 38(3): 181–188, 1995.

[346] Alan Edelman and H. Murakami. Polynomial roots from companion matrix eigenvalues. *Math. Comp.,* 64(210):763–776, 1995.

[347] Alan Edelman and G. W. Stewart. Scaling for orthogonality. *IEEE Trans. Signal Processing,* 41(4):1676-1677, 1993.

[348] Editor's note. *SIAM J. Matrix Anal. Appl.,* 12(3),1991.

[349] Timo Eirola. Aspects of backward error analysis of numerical ODEs. *J. Comp. Appl. Math.,* 45(1-2):65-73, 1993.

[350] Lars Eldén. Perturbation theory for the least squares problem with linear equality constraints. *SIAM J. Numer. Anal.,* 17(3):338–350, 1980.

[351] Samuel K. Eldersveld and Michael A. Saunders. A block-*LU* update for large-scale linear programming. *SIAM J. Matrix Anal. Appl.,* 13(1):191-201, 1992.

[352] W. H. Enright. A new error-control for initial value solvers. *Appl. Math. Comput.,* 31:288-301, 1989.

[353] Michael A. Epton. Methods for the solution of $AXD - BXC = E$ and its application in the numerical solution of implicit ordinary differential equations. *BIT,* 20:341–345, 1980.

[354] A. M. Erisman, R. G. Grimes, J. G. Lewis, W. G. Poole, and H. D. Simon. Evaluation of orderings for unsymmetric sparse matrices. *SIAM J. Sci. Statist. Comput.,* 8(4):600-624, 1987.

[355] A. M. Erisman and J. K. Reid. Monitoring the stability of the triangular factorization of a sparse matrix. *Numer. Math.,* 22:183-186, 1974.

[356] Terje O. Espelid. On floating-point summation. Report No. 67, Department of Applied Mathematics, University of Bergen, Bergen, Norway, December 1978.

[357] Christopher Evans. *Interview with J. H. Wilkinson.* Number 10 in Pioneers of Computing, 60-Minute Recordings of Interviews. Science Museum, London, 1976.

[358] John Ewing, editor. *A Century of Mathematics Through the Eyes of the Monthly*. Mathematical Association of America, Washington, DC, 1994. xi+323 pp. ISBN 0-88385-459-7.

[359] John H. Ewing and F. W. Gehring, editors. *Paul Halmos: Celebrating 50 Years of Mathematics.* Springer-Verlag, New York, 1991. viii+320 pp. ISBN 3-540-97509-8.

[360] V. N. Faddeeva. *Computational Methods of Linear Algebra.* Dover, New York, 1959. x+252 pp. ISBN 0-486-60424-1.

[361] Ky Fan and A. J. Hoffman. Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.,* 6:111-116, 1955.

[362] R. W. Farebrother. A memoir of the life of M. H. Doolittle. *IMA Bulletin,* 23(6/7):102, 1987.

[363] R. W. Farebrother. *Linear Least Sguares Computations.* Marcel Dekker, New York, 1988. xiii+293 pp. ISBN 0-8247-7661-5.

[364] Charles Farnum. Compiler support for floating-point computation. *Software—Practice and Experience,* 18(7):701-709, 1988.

[365] Richard J. Fateman. High-level language implications of the proposed IEEE floating-point standard. *ACM trans. Program. Lang. Syst.,* 4(2):239-257, 1982.

[366] David G. Feingold and Richard S. Varga. Block diagonally dominant matrices and generalizations of the Gerschgorin circle theorem. *Pacific J. Math.,* 12: 1241–1250, 1962.

[367] S. I. Feldman, David M. Gay, Mark W. Maimone, and N. L. Schryer. A Fortran to C converter. Computing Science Technical Report No. 149, AT&T Bell Laboratories, Murray Hill, NJ, USA, 1990. 26 pp.

[368] Alan Feldstein and Richard Goodman. Loss of significance in floating point subtraction and addition. *IEEE Trans. Comput.,* C-31(4):328–335, 1982.

[369] Alan Feldstein and Peter Turner. Overflow, underflow, and severe loss of significance in floating-point addition and subtraction. *IMA J. Numer. Anal.,* 6:241-251, 1986.

[370] Warren E. Ferguson, Jr. Exact computation of a sum or difference with applications to argument reduction. In *Proc. 12th IEEE Symposium on Computer Arithmetic, Bath, England,* Simon Knowles and William H. McAllister, editors, IEEE Computer Society Press, Los Alarnitos, CA, USA, 1995, pages 216–221.

[371] Warren E. Ferguson, Jr. and Tom Brightman. Accurate and monotone approximations of some transcendental functions. In *Pmt. 10th IEEE Symposium on Computer Arithmetic,* Peter Kornerup and David W. Matula, editors, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991, pages 237-244.

[372] William R. Ferng, Gene H. Golub, and Robert J. Plemmons. Adaptive Lanczos methods for recursive condition estimation. *Numer. Algorithms,* l(l): 1–19, 1991.

[373] C. T. Fike. Methods of evaluating polynomial approximations in function evaluation routines. *Comm. ACM,* 10(3):175-178, 1967.

[374] Patrick C. Fischer. Further schemes for combining matrix algorithms. In *Automata, Languages and Programming,* Jacques Loeckx, editor, volume 14 of *Lecture Notes in Computer Science,* Springer-Verlag, Berlin, 1974, pages 428-436.

[375] R. Fletcher. Factorizing symmetric indefinite matrices. *Linear Algebra Appl.,* 14:257–272, 1976.

[376] R. Fletcher, Expected conditioning. *IMA J. Numer. Anal.,* 5:247-273, 1985.

[377] R. Fletcher. Cancellation errors in quasi-Newton methods. *SIAM J. Sci. Statist. Comput.,* 7(4):1387-1399, 1986.

[378] R. Fletcher. *Practical Methods of Optimization.* Second edition, Wiley, Chichester, UK, 1987. xiv+436 pp. ISBN 0-471-91547-5.

[379] R. Fletcher. Degeneracy in the presence of roundoff errors. *Linear Algebra Appl.,* 106:149-183, 1988.

[380] R. Fletcher. Resolving degeneracy in quadratic programming. *Ann. Oper. Res.,* 47:307-334, 1993.

[381] R. Fletcher. Steepest edge, degeneracy and conditioning in LP. Numerical Analysis Report NA/154, Department of Mathematics and Computer Science, University of Dundee, Dundee$_j$ Scotland, November 1994.21 pp.

[382] R. Fletcher and M. J. D. Powell. On the modification of LDL^T factorization. *Math. Comp.,* 28(128):1067-1087, 1974.

[383] Brian Ford. Parameterization of the environment for transportable numerical software. *ACM Trans. Math. Software,* 4(2):100–103, 1978.

[384] Anders Forsgren, Philip E. Gill, and Walter Murray. Computing modified Newton directions using a partial Cholesky factorization. *SIAM J. SCi. Comput.,* 16(1):139-150, 1995.

[385] Anders Forsgren, Philip E. Gill, and Joseph R. Shinnerl. Stability of symmetric ill-conditioned systems arising in interior methods for constrained optimization. *SIAM J. Matrix Anal. Appl.,* 17(1):187–211, 1996.

[386] G. E. Forsythe and E. G. Straus. On best conditioned matrices. *Proc. Amer. Math. Sot.,* 6:340-345, 1955.

[387] George E. Forsythe. Gauss to Gerling on relaxation. *Mathematical Tables and Other Aids to Computation,* 5:255–258, 1951.

[388] George E. Forsythe. Solving linear algebraic equations can be interesting. *Bull. Amer. Math. Sot.,* 59(4):299-329, 1953.

[389] George E. Forsythe. Reprint of a note on rounding-off errors. *SIAM Rev.,* 1 (1):66-67, 1959.

[390] George E. Forsythe. Algorithm 16: Crout with pivoting. *Comm. ACM,* 3(9): 507-508, 1960.

[391] George E. Forsythe. Today's computational methods of linear algebra. *SIAM Rev.,* 9:489-515, 1967.

[392] George E. Forsythe. Solving a quadratic equation on a computer. In *The Mathematical Sciences: A Collection of Essays,* National Research Council's Committee on Support of Research in the Mathematical Sciences, editor, M.I.T. Press, Cambridge, MA, USA, 1969, pages 138-152.

[393] George E. Forsythe. What is a satisfactory quadratic equation solver? In *Constructive Aspects of the Fundamental Theorem of Algebra,* Bruno Dejon and Peter Henrici$_j$ editors, Wiley-Interscience, London, 1969, pages 53–61.

[394] George E. Forsythe. Pitfalls in computation, or why a math book isn't enough. *Amer. Math. Monthly,* 77:931–956, 1970.

[395] George E. Forsythe, Michael A. Malcolm, and Cleve B. Moler. Computer *Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1977. xi+259 pp. ISBN 0-13-165332-6.

[396] George E. Forsythe and Cleve B. Moler. *Computer Solution of Linear Algebraic Systems.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1967. xi+148 pp.

[397] George E. Forsythe and Wolfgang R. Wasow. *Finite-Difference Methods for Partial Differential Equations.* Wiley, New York, 1960. x+444 pp.

[398] Leslie Foster. Modifications of the normal equations method that are numerically stable. In Numerical *Linear Algebra, Digital Signal Processing and Parallel Algorithms, G.* H. Golub and P. M. Van Dooren, editors, volume F70 of *NATO ASI Series,* Springer-Verlag, Berlin, New York, 1991, pages 501-512.

[399] Leslie V. Foster. Gaussian elimination with partial pivoting can fail in practice. *SIAM J. Matrix Anal. Appl.,* 15(4):1354-1362, 1994.

[400] L. Fox. *An Introduction to Numerical Linear Algebra.* Oxford University Press, Oxford, UK, 1964. xi+328 pp.

[401] L. Fox. How to get meaningless answers in scientific computation (and what to do about it). *IMA Bulletin,* 7(10):296-302, 1971.

[402] L. Fox. All about Jim Wilkinson, with a commemorative snippet on backward error analysis. In *The Contribution of Dr. J. H. Wilkinson to Numerical Analysis,* Symposium Proceedings Series No. 19, The Institute of Mathematics and its Applications, Southend-On-Sea, Essex, UK, 1978, pages 1–20.

[403] L. Fox. James Hardy Wilkinson, 1919-1986. *Biographical Memoirs of Fellows of the Royal Society,* 33:671–708, 1987.

[404] L. Fox, H. D. Huskey, and J. H. Wilkinson. Notes on the solution of algebraic linear simultaneous equations. Quart. *J. Mech. Appl. Math.,* 1:149-173, 1948.

[405] P. A. Fox, A. D. Hall, and N. L. Schryer. The PORT mathematical subroutine library. *ACM Trans. Math. Software,* 4(2):104-126, 1978.

[406] Philippe Francois and Jean-Michel Muller. The SCALP perturbation method. In *Proceedings of IMACS '91,* 13*th World Congress on Computation and Applied Mathematics*$_j$ *Dublin, Ireland, Volume* 1, 1991, pages 59-60.

[407] V. Frayssé. *Reliability of Computer Solutions.* PhD thesis, L'Institut National Polytechnique de Toulouse, Toulouse, Fance, July 1992. CERFACS Report TH/PA/92/11.

[408] Shmuel Friedland. Revisiting matrix squaring. *Linear Algebra Appl.,* 154-156: 59-63, 1991.

[409] Shmuel Friedland and Hans Schneider. The growth of powers of a nonnegative matrix. *SIAM J. Alg. Discrete Methods,* 1(2):185–200, 1980.

[410] R. E. Funderlic, M. Neumann, and R. J. Plemmons. *LU* decompositions of generalized diagonally dominant matrices. *Numer. Math.,* 40:57-69, 1982.

[411] Pascal M. Gahinet, Alan J. Laub, Charles S. Kenney, and Gary A. Hewer. Sensitivity of the stable discrete-time Lyapunov equation. *IEEE Trans. Automat. Control,* AC-35(11):1209-1217, 1990.

[412] Shmuel Gal and Boris Bachelis. An accurate elementary mathematical library for the IEEE floating point standard. *ACM Trans. Math. Software,* 17(1): 2645, 1991.

[413] F. R. Gantmacher. The *Theory of Matrices,* volume 1. Chelsea, New York, 1959. x+374 pp. ISBN 0-82840131-4.

[414] F. R. Gantmacher. The *Theory of Matrices,* volume 2. Chelsea, New York, 1959. ix+276 pp. ISBN 0-82840133-0.

[415] B. S. Garbow, J. M. Boyle, J. J. Dongarra, and C. B. Moler. *Matrix Eigensystern Routines—EISPACK Guide Extension,* volume 51 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1977.

[416] Judith D. Gardiner and Alan J. Laub. Parallel algorithms for algebraic Riccati equations. *Int. J. Control,* 54(6):1317-1333, 1991.

[417] Judith D. Gardiner, Alan J. Laub, James J. Amato, and Cleve B. Moler. Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software,* 18(2):223–231, 1992.

[418] Judith D. Gardiner$_j$ Matthew R. Wette, Alan J. Laub, James J. Amato, and Cleve B. Moler. Algorithm 705: A FORTRAN-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Software,* 18(2):232-238, 1992.

[419] Martin Gardner. *More Mathematical Puzzles and Diversions.* Penguin, New York, 1961. 187 pp. ISBN 0-14020748-1.

[420] Harvey L. Garner. A survey of some recent contributions to computer arithmetic. *IEEE Trans. Comput.,* C-25(12):1277–1282, 1976.

[421] M. Gasca and J. M. Peña. Total positivity and Neville elimination. *Linear Algebra Appl.,* 165:25-44, 1992.

[422] Noel Gastinel. *Linear Numerical Analysis.* Kershaw Publishing, London, 1983. ix+341 pp. First published in English by Academic Press, New York, 1970. ISBN 0-901665-16-9.

[423] Carl Friedrich Gauss. *Theory of the Combination of Observations Least Subject to Errors. Part One, Part Two, Supplement.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xi+241 pp. Translated from the Latin and German by G. W. Stewart. ISBN 0-89871-347-1.

[424] Walter Gautschi. On inverses of Vandermonde and confluent Vandermonde matrices. *Numer. Math.,* 4:117–123, 1962,

[425] Walter Gautschi. Norm estimates for inverses of Vandermonde matrices. *Numer. Math.,* 23:337-347, 1975,

[426] Walter Gautschi. On inverses of Vandermonde and confluent Vandermonde matrices III. *Numer. Math.,* 29:445–450, 1978.

[427] Walter Gautschi. The condition of Vandermonde-like matrices involving orthogonal polynomials. *Linear Algebra Appl.,* 52/53:293–300, 1983.

[428] Walter Gautschi. How (un)stable are Vandermonde systems? In *Asymptotic and Computational Analysis,* R. Wong, editor, volume 124 of *Lecture Notes in Pure and Applied Mathematics,* Marcel Dekker, New York and Basel, 1990, pages 193–210.

[429] Walter Gautschi and Gabriele Inglese. Lower bounds for the condition number of Vandermonde matrices. *Numer. Math.,* 52:241-250, 1988.

[430] Werner Gautschi. The asymptotic behaviour of powers of matrices. *Duke Math. J.,* 20:127-140, 1953.

[431] David M. Gay. Correctly rounded binary-decimal and decimal-binary conversions. Numerical Analysis Manuscript 90-10, AT&T Bell Laboratories, Murray Hill, NJ, USA, November 1990. 16 pp.

[432] Stuart Geman. The spectral radius of large random matrices. *Ann. Probab.,* 14(4):1318-1328, 1986.

[433] W. M. Gentleman An error analysis of Goertzel's (Watt's) method for computing Fourier coefficients. *Comput. J.,* 12:160-165, 1969.

[434] W. M. Gentleman and G. Sande. Fast Fourier transforms-for fun and profit. In *Fall* Joint *Computer Conference,* volume 29 of *AFIPS Conference Proceeddings,* Spartan Books, Washington, DC, 1966, pages 563–578.

[435] W. Morven Gentleman. Least squares computations by Givens transformations without square roots. *J. Inst. Maths Applies,* 12:329–336, 1973.

[436] W. Morven Gentleman. Error analysis of *QR* decompositions by Givens transformations. *Linear Algebra Appl.,* 10:189–197, 1975.

[437] W. Morven Gentleman and Scott B. Marovich. More on algorithms that reveal properties of floating point arithmetic units. *Comm. ACM,* 17(5): 276-277, 1974.

[438] Alan George and Joseph W-H Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1981. xii+324 pp. ISBN 0-13-165274-5.

[439] A. J. Geurts. A contribution to the theory of condition. *Numer. Math.,* 39: 85-96, 1982.

[440] Ali R. Ghavimi and Alan J. Laub. Backward error, sensitivity, and refinement of computed solutions of algebraic Riccati equations. Numerical *Linear Algebra with Applications,* 2(1):29-49, 1995.

[441] Ali R. Ghavimi and Alan J. Laub, Residual bounds for discrete-time Lyapunov equations. *IEEE Trans. Automat. Control,* 40(7):1244-1249, 1995.

[442] P. E. Gill and W. Murray. A numerically stable form of the simplex algorithm. *Linear Algebra Appl.,* 7:99-138, 1973.

[443] Philip E. Gill, Walter Murray, Dulce B. Ponceleón, and Michael A. Saunders. Preconditioners for indefinite systems arising in optimization. *SIAM J. Matrix Anal. Appl.,* 13(1):292-311, 1992.

[444] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. User's guide for NPSOL (version 4.0): A Fortran package for nonlinear programming. Technical Report SOL 86-2, Department of Operations Research, Stanford University, Stanford, CA, January 1986. 53 pp.

[445] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. A Schur-complement method for sparse quadratic programming. In *Reliable Numerical Computation,* M. G. Cox and S. J. Hammarling, editors, Oxford University Press, Oxford, UK, 1990, pages 113-138.

[446] Philip E. Gill, Walter Murray, Michael A. Saunders, and Margaret H. Wright. Inertia-controlling methods for general quadratic programming. *SIAM Rev.,* 33(1):1–36, 1991.

[447] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimizetion.* Academic Press, London, 1981. xvi+401 pp. ISBN 0-12-283952-8.

[448] Philip E. Gill, Michael A. Saunders, and Joseph R. Shinnerl. On the stability of Cholesky factorization for symmetric quasi-definite systems. *SIAM J. Matrix Anal. Appl.,* 17(1):35-46, 1996.

[449] S. Gill. A process for the step-by-step integration of differential equations in an automatic digital computing machine. *Proc. Cambridge Phil. Soc.,* 47: 96-108, 1951.

[450] T. A. Gillespie. Noncommutative variations on theorems of Marcel Riesz and others. In *Paul Halmos: Celebrating 50 Years of Mathematics,* John H. Ewing and F. W. Gehring, editors, Springer-Verlag, Berlin, 1991, pages 221–236.

[451] Wallace J. Givens. Numerical computation of the characteristic values of a real symmetric matrix. Technical Report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge, TN, USA, 1954. 107 pp.

[452] James Glanz. Mathematical logic flushes out the bugs in chip designs. *Science,* 267:332–333, 1995. 20 January.

[453] J. Gluchowska and A. Smoktunowicz. Solving the linear least squares problem with very high relative accuracy. *Computing,* 45:345–354, 1990.

[454] I. Gohberg and I. Koltracht. Error analysis for triangular factorization of Cauchy and Vandermonde matrices. Manuscript, 1990.

[455] I. Gohberg and I. Koltracht. Mixed, componentwise, and structured condition numbers. *SIAM J. Matrix Anal. Appl.,* 14(3):688–704, 1993.

[456] I. Gohberg and V. Olshevsky. Fast inversion of Chebyshev–Vandermonde matrices. *Numer. Math.,* 67(1):71–92, 1994.

[457] David Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys,* 23(1):5–48, 1991.

[458] I. Bennett Goldberg. 27 bits are not enough for 8-digit accuracy. *Comm. ACM,* 10(2):105-106, 1967.

[459] Moshe Goldberg and E. G. Straus. Multiplicativity of l_p norms for matrices. *Linear Algebra Appl.,* 52/53:351–360, 1983.

[460] Herman H. Goldstine. *The Computer: From Pascal to von Neumann.* Princeton University Press, Princeton, NJ, USA, 1972. xii+378 pp. 1993 printing with new preface. ISBN 0-691-02367-0.

[461] Herman H. Goldstine. *A History of Numerical Analysis From the 16th Through the 19th Century.* Springer-Verlag, New York, 1977. xiv+348 pp. ISBN 0-387-90277-5.

[462] Herman H. Goldstine and John von Neumann. Numerical inverting of matrices of high order II. *Pmt. Amer. Math. Sot.*, 2:188-202, 1951. Reprinted in [995, pp. 558–572].

[463] G. H. Golub. Numerical methods for solving linear least squares problems. *Numer. Math.*, 7:206–216, 1965.

[464] G. H. Golub, S. Nash, and C. F. Van Loan. A Hessenberg-Schur method for the problem $AX+XB=C$. *IEEE Trans. Automat. Control*, AC-24(6): 909-913, 1979.

[465] G. H. Golub and J. M. Varah. On a characterization of the best ℓ_2-scaling of a matrix. *SIAM J. Numer. Anal.*, 11(3):472–479, 1974.

[466] G. H. Golub and J. H. Wilkinson. Note on the iterative refinement of least squares solution. *Numer. Math.*, 9:139–148, 1966.

[467] G. H. Golub and J. H. Wilkinson. Ill-conditioned eigensystems and the computation of the Jordan canonical form. *SIAM Rev.*, 18(4):578–619, 1976.

[468] Gene H. Golub. Bounds for the round-off errors in the Richardson second order method. *BIT*, 2:212–223, 1962.

[469] Gene H. Golub and Charles F. Van Loan. Unsymmetric positive definite linear systems. *Linear Algebra Appl.*, 28:85-97, 1979.

[470] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Second edition, Johns Hopkins University Press, Baltimore$_j$ MD, USA, 1989. xix+642 pp. ISBN 08018-3772-3 (hardback), 0-8018-3739-1 (paperback).

[471] R. Goodman and A. Feldstein. Round-off error in products. *Computing*, 15: 263-273, 1975.

[472] R. H. Goodman, A. Feldstein, and J. Bustoz. Relative error in floating-point multiplication. *Computing, 35: 127–139, 1985.*

[473] James H. Goodnight. A tutorial on the SWEEP operator. *Amer. Statist.*, 33 (3):149-158, 1979.

[474] N. I. M. Gould. On growth in Gaussian elimination with complete pivoting. *SIAM J. Matrix Anal. Appl.*, 12(2):354-361, 1991.

[475] W. Govaerts and J. D. Pryce. Block elimination with one iterative refinement solves bordered linear systems accurately. *BIT*, 30:490-507, 1990.

[476] W. B. Gragg and G. W. Stewart. A stable variant of the secant method for solving nonlinear equations. *SIAM J. Numer. Anal.*, 13:889-903, 1976.

[477] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science.* Addison-Wesley, Reading, MA, USA, 1989. xiii+625 pp. ISBN 0-201-14236-8.

[478] Andrew Granville. Zaphod Beeblebrox's brain and the fifty-ninth row of Pascal's triangle. *Amer. Math. Monthly*, 99:318-331, 1992.

[479] A. Greenbaum. Behavior of slightly perturbed Lanczos and conjugate-gradient recurrences. *Linear Algebra Appl.,* 113:7-63, 1989.

[480] A. Greenbaum and Z. Strakos. Predicting the behavior of finite precision Lanczos andconjugate gradient computations. *SIAM J. Matrix Anal. Appl.,* 13(1):121-137, 1992.

[481] Anne Greenbaum. The Lanczos andconjugate gradient algorithms infinite precision arithmetic. In *Proceedings of the Cornelius Lanczos International Centenary Conference,* J. David Brown, Moody T. Chu, Donald C. Ellison, and Robert J. Plemmons, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994, pages 49–60.

[482] Robert T. Gregory and David L. Karney. *A Collection of Matrices for Testing Computational Algorithms*. Wiley, New York, 1969. ix+154 pp. Reprinted with corrections by Robert E. Krieger, Huntington, New York, 1978. ISBN 0-88275-649-4.

[483] Roger G. Grimes and John G. Lewis. Condition number estimation for sparse matrices. *SIAM J. Sci. Statist. Comput.,* 2(4):384–388, 1981.

[484] Ming Gu, James W. Demmel, and Inderjit Dhillon. Efficient computation of the singular value decomposition with applications to least squares problems. Technical Report CS-94-257, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, October 1994. 19 pp. LAPACK Working Note 88.

[485] Thorkell Gudmundsson, Charles Kenney, and Alan J. Laub. Small-sample statistical estimates for matrix norms. *SIAM J. Matrix Anal. Appl.,* 16(3): 776–792, 1995.

[486] Heinrich W. Guggenheimer, Alan S. Edelman, and Charles R. Johnson. A simple estimate of the condition number of a linear system. *College Mathematics Journal,* 26(1):2–5, 1995.

[487] Mårten Gulliksson. Iterative refinement for constrained and weighted linear least squares. *BIT,* 34:239-253, 1994.

[488] Mårten Gulliksson. Backward error analysis for the constrained and weighted linear least squares problem when using the weighted *QR* factorization. *SIAM J. Matrix Anal. Appl.,* 16(2):675-687, 1995.

[489] Mårten Gulliksson and Per-Åke Wedin. Modifying the QR-decomposition to constrained and weighted linear least squares. *SIAM J. Matrix Anal. Appl.,* 13(4):1298-1313, 1992.

[490] Chaya Gurwitz. A test for cancellation errors in quasi-Newton methods. *SIAM J. Sci. Statist. Comput.,* 18(2):134-140, 1992.

[491] John L. Gustafson and Srinivas Aluru. Massively parallel searching for better algorithms or, how to do a cross product with five multiplications. *Scientific Programming,* 1996. To appear.

[492] William W. Hager. Condition estimates. *SIAM J. Sci. Statist. Comput.,* 5 (2):311-316, 1984.

[493] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer-Verlag, Berlin, 1991. xv+601 pp. ISBN 3-540-537759.

[494] Marshall Hall, Jr. *Combinatorial Theory.* Blaisdell, Waltham, MA, USA, 1967. x+310 pp.

[495] Hozumi Hamada. A new real number representation and its operation. In *Proceedings of the Eighth* Symposium on *Computer Arithmetic, Como, Italy,* Mary Jane Irwin and Renato Stefanelli, editors, IEEE Computer Society, Washington, DC, 1987, pages 153–157.

[496] S. J. Hammarling. Numerical solution of the stable, non-negative definite Lyapunov equation. *IMA J. Numer. Anal.,* 2:303-323, 1982.

[497] S. J. Hammarling and J. H. Wilkinson. The practical behaviour of linear iterative methods with particular reference to S.O.R. Report NAC 69, National Physical Laboratory, Teddington, UK, September 1976. 19 pp.

[498] Sven Hammarling. A note on modifications to the Givens plane rotation. *J. Inst. Maths Applies,* 13:215-218, 1974.

[499] Stephen M. Hammel, James A. Yorke, and Celso Grebogi. Numerical orbits of chaotic processes represent true orbits. *Bull. Amer. Math. Sot.,* 19(2): 465469, 1988.

[500] Rolf Hammer, Matthias Hocks, Ulrich Kulisch, and Dietmar Ratz. *Numerical Toolbox for Verified Computing I. Basic Numerical Problems: Theory, Algorithms, and Pascal-XSC Programs.* Springer-Verlag, Berlin, 1993. xiii+337 pp. ISBN 3-540-57118-3.

[501] R. W. Hamming. *Numerical Methods for Scientists and Engineers.* Second edition, McGraw-Hill, New York, 1973. ix+721 pp. ISBN 0-07-025887-2.

[502] G. H. Hardy. *A Course of Pure Mathematics.* Tenth edition, Cambridge University Press, Cambridge, UK, 1967. xii+509 pp. ISBN 0-521-09227-2.

[503] G. H. Hardy, J. E. Littlewood, and G. Pólya. *Inequalities.* Second edition, Cambridge University Press, Cambridge, UK, 1952. xii+324 pp.

[504] Richard Harter. The optimality of Winograd’s formula. *Comm. ACM,* 15 (5):352, 1972.

[505] D. J. Hartfiel. Concerning the solution set of $Ax = b$ where $P \leq A \leq Q$ and $p \leq b \leq q$. *Numer. Math.,* 35:355-359, 1980.

[506] *A Manual of Operation for the Automatic Sequence Controlled Calculator.* Harvard University Press, Cambridge, MA, USA, 1946. Reprinted, with new foreword and introduction, Volume 8 in the Charles Babbage Institute Reprint Series for the History of Computing$_j$ MIT Press, Cambridge, MA, USA, 1985. xxxii+561 pp. ISBN 0-262-010844.

[507] *Proceedings of a Symposium on Large-Scale Digital Calculating Machinery,* volume 16 of *The Annals of the Computation Laboratory of Harvard University.* Harvard University Press, Cambridge, MA, USA, 1948. Reprinted, with a new introduction by William Aspray, Volume 7 in the Charles Babbage Institute Reprint Series for the History of Computing, MIT Press, Cambridge, MA, USA, 1985. xxix+302 pp. ISBN 0-262-08152-0.

[508] John Z. Hearon. Nonsingular solutions of $TA - BT = C$. *Linear Algebra Appl.*, 16:57-63, 1977.

[509] M. T. Heath, G. A. Geist, and J. B. Drake. Early experience with the Intel iPSC/860 at Oak Ridge National Laboratory. Report ORNL/TM-11655, Oak Ridge National Laboratory, Oak Ridge, TN, USA, September 1990. 26 pp.

[510] Michael T. Heath. Numerical methods for large sparse linear least squares problems. *SIAM J. Sci. Statist. Comput.*, 5(3):497–513, 1984.

[511] Piet Hein. *Grooks*. Number 85 in Borgen's Pocketbooks. Second edition, Narayana Press, Gylling, 1992. 53 pp. ISBN 87-418-1079-1.

[512] H. V. Henderson and S. R. Searle. On deriving the inverse of a sum. of matrices. *SIAM Rev.*, 23(1):53-60, 1981.

[513] Harold V. Henderson, Friedrich Pukelsheim, and Shayle R. Searle. On the history of the Kronecker product. *Linear and Multilineal Algebra*, 14:113-120, 1983.

[514] Harold V. Henderson and S. R. Searle. The vet-permutation matrix, the vec operator and Kronecker products: A review. *Linear and Multilineal Algebra*, 9:271-288, 1981.

[515] John L. Hennessy and David A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, San Mateo, CA, USA, 1990. xxviii+594+appendices pp. ISBN 1-55860-188-0.

[516] Peter Henrici. Bounds for iterates, inverses, spectral variation and fields of values of non-normal matrices. *Numer. Math.*, 4:24-40, 1962.

[517] Peter Henrici. *Discrete Variable Methods in Ordinary Differential Equations.* John Wiley, New York, 1962. xi+407 pp.

[518] Peter Henrici. *Error Propagation for Difference Methods.* John Wiley, New York, 1963. vi+73 pp.

[519] Peter Henrici. *Elements of Numerical Analysis*. Wiley, New York, 1964. xv+328 pp.

[520] Peter Henrici. Test of probabilistic models for the propagation of roundoff errors. *Comm. ACM*, 9(6):409-410, 1966.

[521] Peter Henrici. A model for the propagation of rounding error in floating arithmetic. In *Interval Mathematics 1980*, Karl L. E. Nickel, editor, Academic Press, New York, 1980, pages 49-73.

[522] Gary Hewer and Charles Kenney. The sensitivity of the stable Lyapunov equation. *SIAM J. Control Optim.*, 26(2):321–344, 1988.

[523] *HP*-15C *Advanced Functions Handbook.* Hewlett-Packard, Portable Computer Division, Corvallis, OR, USA, 1982.221 pp. Part number 00015-90011 Rev. C.

[524] *HP* 48G *Series User's Guide.* Hewlett-Packard, Corvallis Division, Corvallis, OR, USA, 1993. Part number 00048-90126, Edition 3.

[525] Desmond J. Higham. Remark on Algorithm 669. *ACM Trans. Math. Software,* 17(3):424-426, 1991.

[526] Desmond J. Higham. Condition numbers and their condition numbers. *Linear Algebra Appl.,* 214:193-213, 1995.

[527] Desmond J. Higham and Nicholas J. Higham. Backward error and condition of structured linear systems. *SIAM J. Matrix Anal. Appl.,* 13(1):162-175, January 1992.

[528] Desmond J. Higham and Nicholas J. Higham. Componentwise perturbation theory for linear systems with multiple right-hand sides. *Linear Algebra Appl.,* 174:111-129, 1992.

[529] Desmond J. Higham and Lloyd N. Trefethen. Stiffness of ODEs. *BIT,* 33: 285-303, 1993.

[530] Nicholas J. Higharn. Computing the polar decomposition—with applications. *SIAM J. Sci. Statist. Conmput.,* 7(4):1160-1174, 1986.

[531] Nicholas J. Higham. Efficient algorithms for computing the condition number of a tridiagonal matrix. *SIAM J. Sci. Statist. Comput.,* 7(1):150-165, 1986.

[532] Nicholas J. Higham. Computing real square roots of a real matrix. *Linear Algebra Appl.,* 88/89:405-430, 1987.

[533] Nicholas J. Higham. Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems. *Numer. Math.,* 50(5):613-632, 1987.

[534] Nicholas J. Higham. A survey of condition number estimation for triangular matrices. *SIAM Rev.,* 29(4):575–596, 1987.

[535] Nicholas J. Higham. Computing a nearest symmetric positive semidefinite matrix. *Linear Algebra Appl.,* 103:103–118, 1988.

[536] Nicholas J. Higham. Fast solution of Vandermonde-like systems involving orthogonal polynomials. *IMA J. Numer. Anal.,* 8:473-486, 1988.

[537] Nicholas J. Higham. FORTRAN codes for estimating the one-norm of a real or complex matrix, with applications to condition estimation (Algorithm 674). *ACM Trans. Math. Software,* 14(4):381-396, 1988.

[538] Nicholas J. Higham. The accuracy of solutions to triangular systems. *SIAM J. Numer. Anal.,* 26(5):1252-1265, 1989.

[539] Nicholas J. Higham. Matrix nearness problems and applications. In *Applications of Matrix Theory,* M. J. C. Gover and S. Barnett, editors, Oxford University Press, Oxford, UK, 1989, pages 1–27.

[540] Nicholss J. Higham. Analysis of the Cholesky decomposition of a semidefinite matrix. In *Reliable Numerical Computation,* M. G. Cox and S. J, Hammarling, editors, Oxford University Press, Oxford, UK, 1990, pages 161–185.

[541] Nicholas J. Higham. Bounding the error in Gaussian elimination for tridiagonal systems. *SIAM J. Matrix Anal. Appl.,* 11(4):521–530, 1990.

[542] Nicholas J. Higham. Computing error bounds for regression problems. In *Statistical Analysis of Measurement Error Models and Applications, Contemporary Mathematics* 112, Philip J. Brown and Wayne A. Fuller, editors, American Mathematical Society, Providence, RI, USA, 1990, pages 195-208.

[543] Nicholas J. Higham. Experience with a matrix norm estimator. *SIAM J. Sci. Statist. Comput.,* 11(4):804-809, 1990.

[544] Nicholas J. Higham. Exploiting fast matrix multiplication within the level 3 BLAS. *ACM Trans. Math. Software,* 16(4):352-368, 1990.

[545] Nicholas J. Higham. How accurate is Gaussian elimination? In Numerical *Analysis 1989, Proceedings of the* 13*th Dundee Conference,* D. F. Griffiths and G. A. Watson, editors, volume 228 of *Pitman Research Notes in Mathematics,* Longman Scientific and Technical, Essex, UK, 1990, pages 137-154.

[546] Nicholas J. Higham. Iterative refinement enhances the stability of QR factorization methods for solving linear equations. Numerical Analysis Report No. 182, University of Manchester, Manchester, England, April 1990.

[547] Nicholas J. Higham. Stability analysis of algorithms for solving confluent Vandermonde-like systems. *SIAM J. Matrix Anal. Appl.,* 11(1):23-41, 1990.

[548] Nicholas J. Higham. Algorithm 694: A collection of test matrices in MATLAB. *ACM Trans. Math. Software,* 17(3):289-305, 1991.

[549] Nicholas J. Higham. Iterative refinement enhances the stability of QR factorization methods for solving linear equations. *BIT,* 31:447–468, 1991.

[550] Nicholas J. Higham. Three measures of precision in floating point arithmetic. *NA Digest,* Volume 91, Issue 16, 1991. Electronic mail magazine: `na.help@na-net.ornl.gov`.

[551] Nicholas J. Higharn. Estimating the matrix p-norm. *Numer. Math.,* 62: 539-555, 1992.

[552] Nicholas J. Higham. Stability of a method for multiplying complex matrices with three real matrix multiplications. *SIAM J. Matrix Anal. Appl.,* 13(3): 681–687, 1992.

[553] Nicholas J. Higham. The accuracy of floating point summation. *SIAM J. Sci. Comput.,* 14(4):783-799, 1993.

[554] Nicholas J. Higham. *Handbook of Writing for the Mathematical Sciences.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1993. xii+241 pp. ISBN 0-89871-314-5.

[555] Nicholas J. Higham. Optimization by direct search in matrix computations. *SIAM J. Matrix Anal. Appl.,* 14(2):317-333, 1993.

[556] Nicholas J. Higham. Perturbation theory and backward error for $AX - XB = C$. *BIT,* 33:124-136, 1993.

[557] Nicholas J. Higham. The matrix sign decomposition and its relation to the polar decomposition. *Linear Algebra Appl.,* 212/213:3–20, 1994.

[558] Nicholas J. Higham. A survey of componentwise perturbation theory in numerical linear algebra. In *Mathematics of Computation 1943–1993: A Half Century of Computational Mathematics,* Walter Gautschi, editor, volume 48 of *Proceedings of Symposia in Applied Mathematics,* American Mathematical Society, Providence, RI, USA, 1994, pages 49-77.

[559] Nicholas J. Higham. Stability of the diagonal pivoting method with partial pivoting. Numerical Analysis Report No. 265, University of Manchester, Manchester, England, July 1995. 17 pp.

[560] Nicholas J. Higham. Stability of parallel triangular system solvers. *SIAM J. Sci. Comput.,* 16(2):400-413, 1995.

[561] Nicholas J. Higham. The Test Matrix Toolbox for MATLAB, version 3.0. Numerical Analysis Report No. 276, University of Manchester, Manchester, England, September 1995.

[562] Nicholas J. Higham and Desmond J. Higham. Large growth factors in Gaussian elimination with pivoting. *SIAM J. Matrix Anal. Appl.,* 10(2):155–164, 1989.

[563] Nicholas J. Higham and Philip A. Knight. Componentwise error analysis for stationary iterative methods. In *Linear Algebra, Markov Chains, and Queueing Models,* Carl D. Meyer and Robert J. Plemmons, editors, volume 48 of *IMA Volumes in Mathematics and its Applications,* Springer-Verlag, New York, 1993, pages 29-46.

[564] Nicholas J. Higham and Philip A. Knight. Finite precision behavior of stationary iteration for solving singular systems. *Linear Algebra Appl.,* 192: 165–186, 1993.

[565] Nicholas J. Higham and Philip A. Knight. Matrix powers in finite precision arithmetic. *SIAM J. Matrix Anal. Appl.,* 16(2):343–358, 1995.

[566] Nicholas J. Higham and Pythagoras Papadimitriou. A new parallel algorithm for computing the singular value decomposition. In *Proceedings of the Fifth SIAM Conference on Applied Linear Algebra,* John G. Lewis, editor, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1994, pages 80-84.

[567] Nicholas J. Higham and Pythagoras Papadimitriou. A parallel algorithm for computing the polar decomposition. *Parallel Comput.,* 20(8):1161-1173, 1994.

[568] Nicholas J. Higham and Alex Pothen. Stability of the partitioned inverse method for parallel solution of sparse triangular systems. *SIAM J. Sci. Comput.,* 15(1):139-148, 1994.

[569] Nicholas J. Higham and G. W. Stewart. Numerical linear algebra in statistical computing. In *The State of the Art in Numerical Analysis,* A. Iserles and M. J. D. Powell, editors, Oxford University Press, Oxford, UK, 1987, pages 41-57.

[570] David Hilbert. Ein Beitrag zur Theorie des Legendre'schen Polynoms. *Acts Math.,* 18:155–159, 1894.

[571] F. B. Hildebrand. *Introduction to Numerical Analysis.* Second edition, McGraw-Hill, New York, 1974. xiii+669 pp. Reprinted by Dover, New York, 1987. ISBN 0-486-65363-3.

[572] Marlis Hochbruck and Gerhard Starke. Preconditioned Krylov subspace methods for Lyapunov matrix equations. *SIAM J. Matrix Anal. Appl.,* 16 (1):156-171, 1995.

[573] R. W. Hockney and C. R. Jesshope. *Parallel Computers* 2: *Architecture, Programming and Algorithms.* Adam Hilger, Bristol, 1988. xv+625 pp. ISBN 0-85274812-4.

[574] A. Scottedward Hodel. Recent applications of the Lyapunov equation in control theory. In *Iterative Methods in Linear Algebra,* R. Beauwens and P. de Green, editors, Elsevier (North-Holland), Amsterdam, The Netherlands, 1992, pages 217-227.

[575] Andrew Hodges. *Alan Timing: The Enigma.* Burnett Books, London, 1983. 1992 edition with preface, Vintage, London. xix+586 pp. ISBN 0-09-911641-3 .

[576] Christoph M. Hoffmann. The problems of accuracy and robustness in geometric computation. *Computer,* March:31–41, 1989.

[577] W. Hoffmann. Solving linear systems on a vector computer. *J. Comput. Appl. Math.,* 18:353-367, 1987.

[578] W. Hoffmann. Iterative algorithms for Gram-Schmidt orthogonalization. *Computing,* 41:335-348, 1989.

[579] R. C. Holt and J. R. Cordy. The Turing programming language. *Comm. ACM,* 31(12):1410-1423, 1988.

[580] Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1985. xiii+561 pp. ISBN 0-521-30586-1.

[581] Roger A. Horn and Charles R. Johnson. *Topics in Matrix Analysis.* Cambridge University Press, Cambridge, UK, 1991. viii+607 pp. ISBN 0-521-30587-X.

[582] Jim Horning. Note on program reliability. *ACM SIGSOFT Softwam Engineering Notes,* 4(4):6, 1979. Cited in [1021].

[583] Harold Hotelling. Some new methods in matrix calculation. *Ann. Math. Statist.,* 14(1):1-34, 1943.

[584] David Hough. Applications of the proposed IEEE 754 standard for floating-point arithmetic. *Computer,* 14:70-74, 1981.

[585] David Hough. Random story. *NA Digest,* Volume 89, Issue 1, 1989. Electronic mail magazine: `na.help@na-net.ornl.gov.`

[586] Alston S. Householder. Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.,* 5:339-342, 1958.

[587] Alston S. Householder. *The Theory of Matrices in Numerical Analysis.* Blaisdell, New York, 1964. xi+257 pp. Reprinted by Dover, New York, 1975. ISBN 0-486-61781-5.

[588] D. Y. Hu and L. Reichel. Krylov subspace methods for the Sylvester equation. *Linear Algebra Appl.,* 172:283-313, 1992.

[589] T. E. Hull. Correctness of numerical software. In *Perfomance* Evaluation of *Numerical Software,* Lloyd D. Fosdick, editor, North-Holland, Amsterdam, The Netherlands, 1979, pages 3–15.

[590] T. E. Hull. Precision control, exception handling and a choice of numerical algorithms. In *Numerical Analysis Proceedings, Dundee 1981, G.* A. Watson, editor, volume 912 of *Lecture Notes in Mathematics,* Springer-Verlag, Berlin, 1982, pages 169-178.

[591] T. E. Hull, A. Abraham, M. S. Cohen, A. F. X. Curley, C. B. Hall, D. A. Penny, and J. T. M. Sawchuk. Numerical TURING. *ACM SIGNUM Newsletter,* 20(3):26–34, 1985.

[592] T. E. Hull, Thomas F. Fairgrieve, and Ping Tak Peter Tang. Implementing complex elementary functions using except ion handling. *ACM Trans. Math. Software;* 20(2):215-244, 1994.

[593] T. E. Hull and J. R. Swenson. Tests of probabilistic models for propagation of roundoff errors. Comm. *ACM,* 9(2):108–113, 1966.

[594] M. A. Hyman. Eigenvalues and eigenvectors of general matrices. Presented at the 12th National Meeting of the Association for Computing Machinery, Houston, Texas, 1957. Cited in [1088].

[595] IBM. *Engineering and Scientific Subroutine Library, Guide and Reference, Release* 3. Fourth Edition (Program Number 5668-863), 1988.

[596] *AIX Version 3.2 for RISC System\6000: Optimization and Tuning Guide for Fortran, C, and C++.* IBM, December 1993. viii+305 pp. Publication No. SC09-1705-00.

[597] *IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Standard 754-1985.* Institute of Electrical and Electronics Engineers, New York, 1985. Reprinted in SIGPLAN Notices, 22(2):9-25, 1987.

[598] *A Radix-Independent Standard for Floating-Point Arithmetic, IEEE Standard 854-1987.* IEEE Computer Society, New York, 1987.

[599] IEEE Computer Society Microprocessor Standards Committee, Floating-Point Working Group. A proposed standard for binary floating-point arithmetic, Draft 8.0 of IEEE Task P754 (with introductory comments by David Stevenson). *Computer,* 14:51-62, 1981.

[600] Yasuhiko Ikebe. On inverses of Hessenberg matrices. *Linear Algebra Appl.,* 24:93-97, 1979.

[601] ILAS Education Committee. Report on graduate linear algebra courses. Manuscript from the International Linear Algebra Society, November 1993. `URL = http://gauss.technion.ac.il/iic/GRAD-ED.SYLLABI.` 4 pp.

[602] Cray Research Inc. *UNICOS Math and Scientific Library Reference Manual.* Number SR-2081, Version 5.0, Eagan, MN, USA, 1989.

[603] D. C. Ince, editor. *Collected Works of A.M. Turing: Mechanical Intelligence.* North-Holland, Amsterdam, The Netherlands, 1992. xix+227 pp. ISBNO-44488058-5.

[604] F. Incertis. A faster method of computing matrx Pythagorean sums. *IEEE Trans. Automat. Control,* AC-30(3):273-275, 1985.

[605] Ilse C. F. Ipsen and Carl D. Meyer. Uniform stability of Markov chains. *SIAM J. Matrix Anal. Appl.,* 15(4):1061-1074, 1994.

[606] Masao Iri. History of automatic differentiation and rounding error estimation. In *Automatic Differentiation of Algorithms: Theory, Implementation, and Application,* Andreas Griewank and George F. Corliss, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991, pages 3-16.

[607] Eugene Isaacson and Herbert Bishop Keller. *Analysis of Numerical Methods.* Wiley, New York, 1966. xv+541 pp. Reprinted by Dover, New York, 1994. ISBN 0-486-68029-0.

[608] William Jalby and Bernard Philippe. Stability analysis and improvement of the block Gram–Schmidt algorithm. *SIAM J. Sci. Statist. Comput.,* 12(5): 1058–1073, 1991.

[609] M. Jankowski, A. Smoktunowicz, and H. Woźniakowski. A note on floating-point summation of very many terms. *J. Inform. Process. Cybernet.,* 19(9): 435-440, 1983.

[610] M. Jankowski and H.Woźniakowski. Iterative refinement implies numerical stability. *BIT,* 17:303–311, 1977.

[611] M. Jankowski and H. Woźniakowski. The accurate solution of certain continuous problems using only single precision arithmetic. *BIT,* 25:635–651, 1985.

[612] Paul Jansen and Peter Weidner. High-accuracy arithmetic software—some tests of the ACRITH problem-solving routines. *ACM Trans. Math. Software,* 12(1):62–70, 1986.

[613] A. Jennings. Bounds for the singular values of a matrix. *IMA J. Numer. Anal.,* 2:459-474, 1982.

[614] L. S. Jennings and M. R. Osborne. A direct error analysis for least squares. *Numer. Math.,* 22:325-332, 1974.

[615] Mark T. Jones and Merrell L. Patrick. Bunch-Kaufman factorization for real symmetric indefinite banded matrices. *SIAM J. Matrix Anal. Appl.,* 14(2): 553–559, 1993.

[616] Mark T. Jones and Merrell L. Patrick. Factoring symmetric indefinite matrices on high-performance architectures. *SIAM J. Matrix Anal. Appl.,* 15(1): 273–283, 1994.

[617] William B. Jones and W. J. Thron. Numerical stability in evaluating continued fractions. *Math. Comp.,* 28(127):795-810, 1974.

[618] T. L. Jordan. Experiments on error growth associated with some linear least-squares procedures. *Math. Comp.,* 22:579–588, 1968.

[619] George Gheverghese Joseph. *The Crest of the Peacock: Non-European Roots of Mathematics.* Penguin, New York, 1991. xv+371 pp. ISBN 0-14012529-9.

[620] Bo Kågström. A perturbation analysis of the generalized Sylvester equation $(AR - LB, DR - LE) = (C, F)$. *SIAM J. Matrix Anal. Appl.,* 15(4):1045-1060, 1994.

[621] Bo Kågström and Peter Poromaa. Distributed and shared memory block algorithms for the triangular Sylvester equation with sep^{-1} estimators. *SIAM J. Matrix Anal. Appl.,* 13(1):90-101, 1992.

[622] Bo Kågström and Peter Poromaa. LAPACK-style algorithms and software for solving the generalized Sylvester equation and estimating the separation between regular matrix pairs. Report UMINF 93.23, Institute of Information Processing, University of Umeå Umeå, Sweden, December 1993. 35 pp. LAPACK Working Note 75.

[623] Bo Kågström and Peter Poromaa. Computing eigenspaces with specified eigenvalues of a regular matrix pair (A, B) and condition estimation: Theory, algorithms and software. Report UMINF 94.04, Institute of Information Processing, University of Umeå, Umeå, Sweden, September 1994. 65 pp. LAPACK Working Note 87.

[624] Bo Kågström and Lars Westin. Generalized Schur methods with condition estimators for solving the generalized Sylvester equation. *IEEE Trans. Automat. Control* AC-34(7):745-751, 1989.

[625] W. Kahan Further remarks on reducing truncation errors. *Comm. ACM,* 8 (1):40, 1965.

[626] W. Kahan. Numerical linear algebra. *Canad. Math. Bull.,* 9:757-801, 1966.

[627] W. Kahan. A survey of error analysis. In *Proc. IFIP Congress, Ljubijana,* Information Processing 71, North-Holland, Amsterdam, The Netherlands, 1972, pages 1214-1239.

[628] W. Kahan. Implementation of algorithms (lecture notes by W. S. Haugeland and D. Hough). Technical Report 20, Department of Computer Science, University of California, Berkeley, CA, USA, 1973.

[629] W. Kahan. Interval arithmetic options in the proposed IEEE floating point arithmetic standard. In *Interval Mathematics 1980,* Karl L. E. Nickel, editor, Academic Press, New York, 1980, pages 99-128.

[630] W. Kahan. Why do we need a floating-point arithmetic standard? Technical report, University of California, Berkeley, CA, USA, February 1981, 41 pp.

[631] W. Kahan. To solve a real cubic equation. Technical Report PAM-352, Center for Pure and Applied Mathematics, University of California, Berkeley, CA, USA, November 1986. 20 pp.

[632] W. Kahan. Branch cuts for complex elementary functions or much ado about nothing's sign bit. In *The State of the Art in Numerical Analysis,* A. Iserles

and M. J. D. Powell, editors, Oxford University Press, Oxford, UK, 1987, pages 165–21 1.

[633] W. Kahan. Doubled-precision IEEE standard 754 floating-point arithmetic. Manuscript, February 1987.

[634] W. Kahan. How Cray's arithmetic hurts scientific computation (and what might be done about it). Manuscript prepared for the Cray User Group meeting in Toronto, June 1990. *42* pp.

[635] W. Kahan. Analysis and refutation of the LCAS. *ACM SIGNUM Newsletter,* 26(3):2–15, 1991.

[636] W. Kahan. Computer benchmarks versus accuracy. Draft manuscript, June 1994.

[637] W. Kahan and I. Farkas.. Algorithm 168: Newton interpolation with backward divided differences. *Comm. ACM,* 6(4):165, 1963.

[638] W. Kahan and I. Farkas. Algorithm 169: Newton interpolation with forward divided differences. *Comm. ACM,* 6(4):165, 1963.

[639] W. Kahan and E. LeBlanc. Anomalies in the IBM ACRITH package. In *Proceedings of the 7th Symposium on Computer Arithmetic,* Kai Hwang, editor, IEEE Computer Society Press, Silver Spring, MD, USA, 1985, pages 322-331.

[640] W. Kahan and J. Palmer. On a proposed floating-point standard. *ACM SIGNUM Newsletter,* 14:13-21, October 1979.

[641] David K. Kahaner, Cleve B. Moler, and Stephen G. Nash. Numerical *Methods and Software.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1989. xii+495 pp. ISBN 0-13-627258-4.

[642] Ilkka Karasalo. A criterion for truncation of the *QR*-decomposition algorithm for the singular linear least squares problem. *BIT,* 14:156-166, 1974.

[643] A. Karatsuba and Yu. Ofman. Multiplication of multidigit numbers on automata. *Soviet Phys. Dokl.,* 7(7):595–596, January 1963.

[644] Samuel Karlin. *Total Positivity,* volume 1. Stanford University Press, Stanford, CA, USA, 1968.

[645] Richard Karpinski. Paranoia: A floating-point benchmark. *BYTE,* 10(2): 223-235, 1985.

[646] Tosio Kate. *Perturbation Theory for Linear Operators.* Second edition, Springer-Verlag, Berlin, 1976. xxi+619 pp. ISBN 3-540-97588-5.

[647] Linda Kaufman. Matrix methods for queuing problems. *SIAM J. Sci. Statist. Comput.,* 4(3):525-552, 1983.

[648] Herbert B. Keller. On the solution of singular and semidefinite linear systems by iteration. *SIAM J. Numer. Anal.,* 2(2):281-290, 1965.

[649] William J. Kennedy, Jr. and James E. Gentle. *Statistical Computing.* Marcel Dekker, New York, 1980. xi+591 pp. ISBN 0-8247-6898-1.

[650] Charles Kenney and Gary Hewer. The sensitivity of the algebraic and differential Riccati equations. *SIAM J. Control Optim.,* 28(1):50-69, 1990.

[651] Charles Kenney and Alan J. Laub. Controllability and stability radii for companion form systems. *Math. Control Signals Systems,* 1:239-256, 1988.

[652] Charles S. Kenney and Alan J. Laub. Small-sample statistical condition estimates for general matrix functions. *SIAM J. Sci. Comput.,* 15(1):36-61, 1994.

[653] Charles S. Kenney, Alan J. Laub, and Philip M. Papadopoulos. Matrix sign algorithms for Riccati equations. *IMA J. of Math. Control Inform.,* 9:331-344, 1992.

[654] Thomas H. Kerr. Fallacies in computational testing of matrix positive definiteness/semidefiniteness. *IBEE Trans. Aerospace Electron. Systems,* 26(2):415–421, 1990.

[655] Andrzej Kiełbasiński. Summation algorithm with corrections and some of its applications. *Math. Stos.,* 1:22–41, 1973. (In Polish, cited in [609] and [611]).

[656] Andrzej Kiełbasiński. Iterative refinement for linear systems in variable precision arithmetic. *BIT,* 21:97-103, 1981.

[657] Andrzej Kiełbasiński A note on rounding-error analysis of Cholesky factorization. *Linear Algebra Appl.,* 88/89:487–494, 1987.

[658] Andrzej Kiełbasiński and Hubert Schwetlick. *Numerische Lineare Algebra: Eine Computerorientierte Einführung.* VEB Deutscher, Berlin, 1988.472 pp. ISBN 3-87144999-7.

[659] Andrzej Kiełbasiński and Hubert Schwetlick. *Numeryczna Algebra Liniowa: Wprowadzenie doObliczeń Zautomatyzowanych.* Wydawnictwa Naukowo-Techniczne, Warszawa, 1992. 502 pp. ISBN 83-2041260-9.

[660] Fuad Kittaneh. Singular values of companion matrices and bounds on zeros of polynomials. *SIAM J. Matrix Anal. Appl.,* 16(1):333–340, 1995.

[661] R. Klatte, U. W. Kulisch, C. Lawo, M. Rauch, and A. Wiethoff. *C-XSC: A C++ Class Library for Extended Scientific Computing.* Springer-Verlag, Berlin, 1993. ISBN 0-387-56328-8.

[662] R. Klatte, U. W. Kulisch, M. Neaga, D. Ratz, and Ch. Ullrich. *PASCAL-XSC—Language Reference With Examples.* Springer-Verlag, Berlin, 1992.

[663] Philip A. Knight. *Error Analysis of Stationary Iteration and Associated Problems,* Ph.D. thesis, University of Manchester, Manchester, England, September 1993. 135 pp.

[664] Philip A. Knight. Fast rectangular matrix multiplication and *QR* decomposition. *Linear Algebra Appl.,* 221:69–81, 1995.

[665] Donald E. Knuth. Evaluation of polynomials by computer. *Comm. ACM,* 5 (12):595-599, 1962.

[666] Donald E. Knuth. *The Art of Computer Programming.* Addison-Wesley, Reading, MA, USA, 1973-1981. Three volumes.

[667] Donald E. Knuth. *The Art of Computer Programming, Volume* 1, *Fundamental Algorithms.* Second edition, Addison-Wesley, Reading, MA, USA, 1973. xxi+634 pp. ISBN 0-201-03821-8.

[668] Donald E. Knuth. *The Art of Computer Programming, Volume 2, Seminumerical Algorithms.* Second edition, Addison-Wesley, Reading, MA, USA, 1981. xiii+688 pp. ISBN 0-201-03822-6.

[669] Donald E. Knuth. Two notes on notation. *Amer. Math. Monthly,* 99(5): 403-422, 1992.

[670] T. W. Körner. *Fourier Analysis.* Cambridge University Press, Cambridge, UK, 1988. xii+591 pp. ISBN 0521389917.

[671] Eric Kostlan. On the spectra of Gaussian matrices. *Linear Algebra Appl.,* 162-164:385–388, 1992.

[672] Z. V. Kovarik. Compatibility of approximate solutions of inaccurate linear equations. *Linear Algebra Appl.,* 15:217–225, 1976.

[673] Antoni Kreczmar. On memory requirements of Strassen's algorithms. volume 45 of *Lecture Notes in Computer Science,* Springer-Verlag, Berlin, 1976, pages 404–407.

[674] Ed Krol. *The Whole Internet User's Guide & Catalog.* Second edition, O'Reilly & Associates, Sebsstopol, CA, USA, 1994. xxv+543 pp. ISBN 1-56592-063-5.

[675] Koichi Kubota. PADRE2, a Fortran precompiled yielding error estimates and second derivatives. In *Automatic Differentiation of Algorithms: Theory, Implementation, and Application,* Andreas Griewank and George F. Corliss, editors, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991, pages 251-262.

[676] J. Kuczyński and H. Woźniakowski. Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. *SIAM J. Matrix Anal. Appl.,* 13(4):1094-1122, 1992.

[677] H. Kuki and W. J. Cody. A statistical study of the accuracy of floating point number systems. *Comm. ACM,* 16(4):223–230, 1973.

[678] Ulrich W. Kulisch and Willard L. Miranker. *Computer Arithmetic in Theory and in Practice.* Academic Press, New York, 1981. xiii+249 pp. ISBN 0-12-428650-X.

[679] Ulrich W. Kulisch and Willard L. Miranker, editors. *A New Approach to Scientific Computation.* Academic Press, New York, 1983. xv+384 pp. ISBN 0-12-428660-7.

[680] Ulrich W. Kulisch and Willard L. Miranker. The arithmetic of the digital computer: A new approach. *SIAM Rev.,* 28(1):1–40, 1986.

[681] I. B. Kuperman. *Approximate Linear Algebraic Equations.* Van Nostrand Reinhold, London, 1971. xi+225 pp. ISBN 0-442-04546-8.

[682] M. La Porte and J. Vignes. Etude statistique des erreurs clans l'arithmétique des ordinateurs; application au contrôe des résultats d'algorithmes numériques. *Numer, Math.,* 23:63–72, 1974.

[683] J. D. Laderman. A noncommutative algorithm for multiplying 3 × 3 matrices using 23 multiplications. *Bull. Amer. Math. Soc.,* 82(1):126–128, 1976.

[684] Julian Laderman, Victor Pan, and Xuan-He Sha. On practical algorithms for accelerated matrix multiplication. *Linear Algebra Appl.,* 162-164:557–588, 1992.

[685] Peter Lancaster. Explicit solutions of linear matrix equations. *SIAM Rev.,* 12(4):544–566, 1970.

[686] Cornelius Lanczos. *Applied Analysis.* Prentice Hall, Englewood Cliffs, NJ, USA, 1956. xx+539 pp. Reprinted by Dover, New York, 1988. ISBN 0-486-65656-X.

[687] John L. Larson, Mary E. Pasternak, and John A. Wisniewski. Algorithm 594: Software for relative error analysis. *ACM Trans. Math. Software,* 9(l): 125–130, 1983.

[688] John L. Larson and Ahmed H. Sameh. Efficient calculation of the effects of roundoff errors. *ACM Trans. Math. Software,* 4(3):228–236, 1978. Errata 5(3):372, 1979.

[689] John L. Larson and Ahmed H. Sameh. Algorithms for roundoff error analysis—A relative error approach. *Computing,* 24:275–297, 1980.

[690] Lajos László. An attainable lower bound for the best normal approximation. *SIAM J. Matrix Anal. Appl.,* 15(3):1035-1043, 1994.

[691] Alan J. Laub. A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automat. Control,* AC-24(6):913-921, 1979.

[692] Peter Läuchli. Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. *Numer. Math.,* 3:226-240, 1961.

[693] Simon Lavington. *Early British Computers: The Story of Vintage Computers and the People Who Built Them.* Manchester University Press, 1980. 139 pp. ISBN 0-7190-0803-4.

[694] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Math. Software,* 5(3): 308-323, 1979.

[695] Charles L. Lawson and Richard J. Hanson. *Solving Least Squares Problems.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1995. xii+340 pp. Corrected republication of work first published in 1974 by Prentice-Hall. ISBN 0-89871-356-0.

[696] Lam Lay-Yong and Shen Kangshen. Methods of solving linear equations in traditional China. *Historia Mathematical,* 16(2):107–122, 1989.

[697] D. H. Lehmer. Tables to many places of decimals. *Mathematical Tables and Other Aids to Computation,* 1(1):30-31, 1943.

[698] R. B. Lehoucq. The computation of elementary unitary matrices. Technical Report CS-94-233, Department of Computer Science, University of Tennessee, Knoxville, TN, USA, May 1994. 9 pp. LAPACK Working Note 72.

[699] Frans Lemeire. Bounds for condition numbers of triangular and trapezoid matrices. *BIT,* 15:58–64, 1975.

[700] H. Leuprecht and W. Oberaigner. Parallel algorithms for the rounding exact summation of floating point numbers. *Computing,* 28:89-104, 1982.

[701] T. Y. Li and Z. Zeng. Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems. *Math. Comp.,* 59(200):483–502, 1992.

[702] Information Technology-Language Independent Arithmetic—Part I: Integer and Floating Point Arithmetic, Draft International Standard (Version 4.1), ISO/IEC DIS 10967-1:1993. August 1993.

[703] Seppo Linnainmaa. Analysis of some known methods of improving the accuracy of floating-point sums. *BIT,* 14:167–202, 1974.

[704] Seppo Linnainmaa. Towards accurate statistical estimation of rounding errors in floating-point computations. *BIT,* 15:165-173, 1975.

[705] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. *BIT,* 16:146-160, 1976.

[706] Seppo Linnainmaa. Software for doubled-precision floating-point computations. *ACM Trans. Math. Software,* 7(3):272–283, 1981.

[707] Peter Linz. Accurate floating-point summation. *Comm. ACM,* 13(6):361–362, 1970.

[708] Elliot Linzer. On the stability of transform-based circular deconvolution. *SIAM J. Numer. Anal.,* 29(5):1482-1492, 1992.

[709] Joseph W. H. Liu. A partial pivoting strategy for sparse symmetric matrix decomposition. *ACM Trans. Math. Software,* 13(2):173-182, 1987.

[710] Georghios Loizou. Nonnormality and Jordan condition numbers of matrices. *J. Assoc. Comput. Mach.,* 16(4):580-584, 1969.

[711] J. W. Longley. An appraisal of least squares programs for the electronic computer from the point of view of the user. *J. Amer. Statist. Assoc.,* 62: 819-841, 1967.

[712] Per Lotstedt. Perturbation bounds for the linear least squares problem subject to linear inequality constraints. *BIT,* 23:500-519, 1983.

[713] Per Lotstedt. Solving the minimal least squares problem subject to bounds on the variables. *BIT,* 24:206-224, 1984.

[714] Hao Lu. Fast solution of confluent Vandermonde linear systems. *SIAM J. Matrix Anal. Appl.,* 15(4):1277-1289, 1994.

[715] Hao Lu. Fast algorithms for confluent Vandermonde linear systems and generalized Trummer's problem. *SIAM J. Matrix Anal. Appl.,* 16(2):655-674, 1995.

[716] Hao Lu. Solution of Vandermond-like systems and confluent Vandermonde-like systems. *SIAM J. Matrix Anal. Appl.,* 17(1):127-138, 1996.

[717] J. N. Lyness. The effect of inadequate convergence criteria in automatic routines. *Computer Journal,* 12:279-281, 1969. See also letter and response in 13 (1970), p. 121.

[718] J. N. Lyness and C. B. Moler. Van der Monde systems and numerical differentiation. *Numer. Math.,* 8:458–464, 1966.

[719] M. Stuart Lynn. On the round-off error in the method of successive over-relaxation. *Math. Comp.,* 18(85):36–49, 1964.

[720] Allan J. Macleod. Some statistics on Gaussian elimination with partial pivoting. *ACM SIGNUM Newsletter,* 24(2/3):10-14, 1989.

[721] J. H. Maindonald. *Statistical Computation.* Wiley, New York, 1984. xviii+370 pp. ISBN 0-471-86452-8.

[722] John Makhoul. Toeplitz determinants and positive semidefiniteness. *IEEE Trans. Signal Processing,* 39(3):743-746, 1991.

[723] Michael A. Malcolm. On accurate floating-point summation. *Comm. ACM,* 14(11):731-736, 1971.

[724] Michael A. Malcolm. Algorithms to reveal properties of floating-point arithmetic. *Comm. ACM,* 15(11):949–951, 1972.

[725] Michael A. Malcolm and John Palmer. A fast method for solving a class of tridiagonal linear systems. *Comm. ACM,* 17(1):14–17, 1974.

[726] Thomas A. Manteuffel. An interval analysis approach to rank determination in linear least squares problems. *SIAM J. Sci. Statist. Comput.,* 2(3):335–348, 1981.

[727] John Markoff. Circuit flaw causes Pentium chip to miscalculate, Intel admits. *New York Times,* 1994. 24 November.

[728] George Marsaglia and Ingram Olkin. Generating correlation matrices. *SIAM J. Sci. Statist. Comput.,* 5(2):470-475, 1984.

[729] R. S. Martin, G. Peters, and J. H. Wilkinson. Iterative refinement of the solution of a positive definite system of equations. *Numer. Math.,* 8:203–216, 1966. Also in [1102, pp. 31–44], Contribution 1/2.

[730] Gleanings far and near. *Mathematical Gazette,* 22(170):95, 1924.

[731] Roy Mathias. Matrices with positive definite Hermitiam part: Inequalities and linear systems. *SIAM J. Matrix Anal. Appl.,* 13(2):640-654, 1992.

[732] Roy Mathias. Accurate eigensystem computations by Jacobi methods. *SIAM J. Matrix Anal. Appl.,* 16(3):977-1003, 1995.

[733] Roy Mathias. Analysis of algorithms for orthogonalizing products of unitary matrices. *Numerical Linear Algebra with Applications,* 1995. To appear.

[734] Roy Mathias. The instability of parallel prefix matrix multiplication. *SIAM J. Sci. Comput.,* 16(4):956-973, 1995.

[735] *MATLAB User's Guide.* The MathWorks, Inc., Natick, MA, USA, 1992.

[736] Shouichi Matsui and Masao Iri. An overflow/underflow-free floating-point representation of numbers. *J. Inform. Process.,* 4(3):123–133, 1981.

[737] R. M. M. Mattheij. Stability of block *LU*-decompositions of matrices arising from BVP. *SIAM J. Alg. Discrete Methods,* 5(3):314-331, 1984.

[738] R. M. M. Mattheij. The stability of LU-decompositions of block tridiagonal matrices. *Bull. Austral. Math. Soc.,* 29:177–205, 1984.

[739] David W. Matula. In-and-out conversions. *Comm. ACM,* 11(1):47-50, 1968.

[740] David W. Matula. A formalization of floating-point numeric base conversion. *IEEE Trans. Comput.,* C-19(8):681-692, 1970.

[741] David W. Matula and Peter Kornerup. Finite precision rational arithmetic: Slash number systems. *IEEE Trans. Comput.,* C-34(1):3-18, 1985.

[742] Charles McCarthy and Gilbert Strang. Optimal conditioning of matrices. *SIAM J. Numer. Anal.,* 10(2):370-388, 1974.

[743] Daniel D. McCracken and William S. Dorn. Numerical *Methods and Fortran Programming: With Applications in Science and Engineering.* Wiley, New York, 1964. xii+457 pp.

[744] P. McCullagh and J. A. Nelder. *Generalized Linear Models.* Second edition, Chapman and Hall, London, 1989. xix+511 pp. ISBN 0-412-31760-5.

[745] William Marshall McKeeman. Algorithm 135: Crout with equilibration and iteration. *Comm. ACM,* 5:553–555, 1962.

[746] Jean Meinguet. On the estimation of significance. In Topics in *Interval Analysis,* Elden Hansen, editor, Oxford University Press, Oxford, UK, 1969, pages 47–64.

[747] Jean Meinguet. Refined error analyses of Cholesky factorization. *SIAM J. Numer. Anal.,* 20(6):1243-1250, 1983.

[748] N. S. Mendelssohn. Some elementary properties of ill conditioned matrices and linear equations. *Amer. Math. Monthly,* 63(5):285–295, 1956.

[749] Michael Metcalf and John K. Reid. *Fortran* 90 *Explained.* Oxford University Press, Oxford, UK, 1990. xiv+294 pp. ISBN 0-19-853772-7.

[750] N. Metropolis. Methods of significance arithmetic. In *The State of the Art in Numerical Analysis,* David A. H. Jacobs, editor, Academic Press, London, 1977, pages 179-192.

[751] Gérard Meurant. A review on the inverse of symmetric tridiagonal and block tridiagonal matrices. *SIAM J. Matrix Anal. Appl.,* 13(3):707–728, 1992.

[752] Carl D. Meyer, Jr. and R. J. Plemmons. Convergent powers of a matrix with applications to iterative methods for singular linear systems. *SIAM J. Numer. Anal.,* 14(4):699-705, 1977.

[753] H. I. Meyer and B. J. Hollingsworth. A method of inverting large matrices of special form. *M.T.A.C.,* 11:94–97, 1957.

[754] Victor J. Milenkovic. Verifiable implementations of geometric algorithms using finite precision arithmetic. *Artif. Intell.,* 37:377–401, 1988.

[755] D. F. Miller. The iterative solution of the matrix equation $XA+BX+C = 0$. *Linear Algebra Appl.,* 105:131-137, 1988.

[756] Webb Miller. Computational complexity and numerical stability. *SIAM J. Comput.,* 4(2):97-107, 1975.

[757] Webb Miller. Software for roundoff analysis. *ACM Trans. Math. Software,* 1 (2):108-128, 1975.

[758] Webb Miller. Graph transformations for roundoff analysis. *SIAM J. Comput.,* 5(2):204–216, 1976.

[759] Webb Miller. *The Engineering of Numerical Software.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1984. viii+167 pp. ISBN 0-13-279043-2.

[760] Webb Miller and David Spooner. Software for roundoff analysis, II. *ACM Trans. Math. Software,* 4(4):369-387, 1978.

[761] Webb Miller and David Spooner. Algorithm 532: Software for roundoff analysis. *ACM Trans. Math. Software,* 4(4):388-390, 1978.

[762] Webb Miller and Celia Wrathall. *Software for Roundoff Analysis of Matrix Algorithms.* Academic Press, New York, 1980. x+151 pp. ISBN 0-12-497250-0.

[763] L. Mirsky. *An Introduction to Linear Algebra.* Oxford University Press, 1961. viii+440 pp. Reprinted by Dover, New York, 1990. ISBN 0-486-66434-1.

[764] Herbert F. Mitchell, Jr. Inversion of a matrix of order 38. *M.T.A.C.,* 3: 161-166, 1948.

[765] Cleve B. Moler. Iterative refinement in floating point. *J. Assoc. Comput. Mach.,* 14(2):316-321, 1967.

[766] Cleve B. Moler. Matrix computations with Fortran and paging. *Comm. ACM,* 15(4):268-270, 1972.

[767] Cleve B. Moler. Algorithm 423: Linear equation solver. *Comm. ACM,* 15 (4):274, 1972.

[768] Cleve B. Moler. Cramer's rule on 2-by-2 systems. *ACM SIGNUM Newsletter,* 9(4):13-14, 1974.

[769] Cleve B. Moler. Three research problems in numerical linear algebra. In *Numerical Analysis,* G. H. Golub and J. Oliger, editors, volume 22 of *Proceedings of Symposia in Applied Mathematics,* American Mathematical Society, Providence, RI, USA, 1978, pages 1-18.

[770] Cleve B. Moler. Cleve's corner: The world's simplest impossible problem. *The Math Works Newsletter,* 4(2):6-7, 1990.

[771] Cleve B. Moler. Technical note: Doubl-rounding and implications for numeric computations. *The Math Works Newsletter,* 4:6, 1990.

[772] Cleve B. Moler. A tale of two numbers. *SIAM News,* 28:1,16, January 1995. Also in MATLAB *News and Notes,* Winter 1995, 10-12.

[773] Cleve B. Moler and Donald Morrison. Replacing square roots by Pythagorean sums. *IBM J. Res. Develop.,* 27(6):577–581, 1983.

[774] Cleve B. Moler and Donald Morrison. Singular value analysis of cryptograms. *Amer. Math. Monthly,* 90:78-87, 1983.

[775] Cleve B. Moler and Charles F. Van Loan. Nineteen dubious ways to compute the exponential of a matrix. *SIAM Rev.,* 20(4):801–836, 1978.

[776] Ole Møller. Note on quasi double-precision. *BIT*, 5:251-255, 1965.

[777] Ole Møller. Quasi double-precision in floating point addition. *BIT*, 5:37-50, 1965. See also [776] for remarks on this article.

[778] Ramon E. Moore. *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1966. xi+145 pp.

[779] Ramon E. Moore. *Methods and Applications of Interval Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1979. xi+190 pp. ISBN 0-89871-161-4.

[780] Robb J. Muirhead. *Aspects of Multivariate Statistical Theory*. Wiley, New York, 1982. xix+673 pp. ISBN 0-471-09442-0.

[781] Jean-Michel Muller. *Arithmétigue des Ordinateurs*. Masson, Paris, 1989.214 pp. In French. Cited in [406]. ISBN 2-225-81689-1.

[782] K. H. Müller. Rounding error analysis of Homer's scheme. *Computing*, 30: 285-303, 1983.

[783] H. Müller-Merbach. *On Round-Off Errors in Linear Programming*, volume 37 of *Lecture Notes in Operations Research and Mathematical Systems*. Springer-Verlag, Berlin, 1970. 48 pp.

[784] FPV: A floating-point validation package. Release I. User's guide. Technical Report NP 1201, NAG Ltd., Oxford, UK, May 1986.

[785] *NAGWare FTN90 Reference Manual*. NAG Ltd., Oxford, UK, 1992. ISBN 1-85206-080-8.

[786] M. Zuhair Nashed and L. B. Rail. Annotated bibliography on generalized inverses and applications. In Generalized *Inverses and Applications*, M. Zuhair Nashed, editor, Academic Press, New York, 1976, pages 771-1041.

[787] J. A. Nelder and R. Mead. A simplex method for function minimization. *Comput. J.*, 7:308-313, 1965.

[788] A. Neumaier. Rundungsfehleranalyse einiger Verfahren zur Summation endlicher Summen. Z. *Angew. Math. Mech.*, 54:39–51, 1974.

[789] A. Neumaier. Inner product rounding error analysis in the presence of underflow. *Computing*, 34:365-373, 1985.

[790] A. Neumaier. On the comparison of H-matrices with M-matrices. *Linear Algebm Appl.*, 83:135-141, 1986.

[791] M. Neumann and R. J. Plemmons. Backward error analysis for linear systems associated with inverses of H-matrices. *BIT*, 24:102–112, 1984.

[792] A. C. R. Newbery. Error analysis for Fourier series evaluation. *Math. Comp.*, 27(123):639-644, 1973.

[793] A. C. R. Newbery. Error analysis for polynomial evaluation. *Math. Comp.*, 28(127):789-793, 1974.

[794] Simon Newcomb. Note on the frequency of use of the different digits in natural numbers. *Amer. J. Math.*, 4:39–40, 1881. Cited in [856].

[795] Morris Newman. Matrix computations. In *Survey of Numerical Analysis,* John Todd, editor, McGraw-Hill, New York, 1962, pages 222-254.

[796] Morris Newman and John Todd. The evaluation of matrix inversion programs. *J. Sot. Indust. Appl. Math.,* 6(4):466-476, 1958.

[797] K. Nickel. Das Kahan-Babuškasche Summierungsverfahren in Triplex-ALGOL 60. *Z. Angew. Math. Mech.,* 50:369-373, 1970.

[798] Karl Nickel. Interval-analysis. In *The State of the Art in Numerical Analysis,* David A. H. Jacobs, editor, Academic Press, London, 1977, pages 193-225.

[799] Yvan Notay. On the convergence rate of the conjugate gradients in presence of rounding errors. *Numer. Math.,* 65:301–317, 1993.

[800] Colm Art O'Cinneide. Entrywise perturbation theory and error analysis for Markov chains. Numer. *Math.,* 65:109-120, 1993.

[801] W. Oettli. On the solution set of a linear system with inaccurate coefficients. *SIAM J. Numer. Anal.,* 2(1):115-118, 1965.

[802] W. Oettli and W. Prager. Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math.,* 6:405-409, 1964.

[803] W. Oettli, W. Prager, and J. H. Wilkinson. Admissible solutions of linear systems with not sharply defined coefficients. *SIAM J. Numer. Anal,* 2(2): 291-299, 1965.

[804] Dianne Prost O'Leary. Estimating matrix condition numbers. *SIAM J. Sci. Statist. Comput.,* 1(2):205-209, 1980.

[805] J. Oliver. An error analysis of the modified Clenshaw method for evaluating Chebyshev and Fourier series. *J. Inst. Maths Applies,* 20:379-391, 1977.

[806] J. Oliver. Rounding error propagation in polynomial evaluation schemes. *J. Comput. Appl. Math.,* 5(2):85-97, 1979.

[807] F. W. J. Olver. A new approach to error arithmetic. *SIAM J. Numer. Anal.,* 15(2):368-393, 1978.

[808] F. W. J. Olver. Error analysis of complex arithmetic. In *Computational Aspects of Complex Analysis,* volume 102 of *NATO Advanced Study Institute Series C,* D. Reidel, editor, Dordrecht, Holland, 1983, pages 279-292.

[809] F. W. J. Olver. Error bounds for polynomial evaluation and complex arithmetic. *IMA J. Numer. Anal.,* 6:373–379, 1986.

[810] F. W. J. Olver and J. H. Wilkinson. *A posteriori* error bounds for Gaussian elimination. *IMA J. Numer. Anal.,* 2:377–406, 1982.

[811] James M. Ortega. An error analysis of Householder's method for the symmetric eigenvalue problem. *Numer. Math.,* 5:211-225, 1963.

[812] A. M. Ostrowski. *Solution of Equations in Euclidean and Banach Spaces.* Academic Press, New York, 1973. xx+412 pp. Third edition of Solution of Equations and Systems of Equations. ISBN 0-12530260-6.

[813] C. C. Paige. An error analysis of a method for solving matrix equations. *Math. Comp.,* 27(122):355-359, 1973.

[814] C. C. Paigeand M. Wei. History andgenerality of the CS decomposition. *Linear Algebra Appl.,* 208/209:303-326, 1994.

[815] V. Pan. Strassen algorithm is not optimal. Trilinear technique of aggregating, uniting and canceling for constructing fast algorithms for matrix multiplication. In *Proc. 19th Annual Symposium on the Foundations of Computer Science,* Ann Arbor, MI, USA, 1978, pages 166-176.

[816] Victor Pan. How can we speed up matrix multiplication? *SIAM Rev.,* 26(3): 393-415, 1984.

[817] Beresford N. Parlett. Laguerre's method applied to the matrix eigenvalue problem. *Math. Comp.,* 18:464-485, 1964.

[818] Beresford N. Parlett. Matrix eigenvalue problems. *Amer. Math. Monthly,* 72 (2):59-66, 1965.

[819] Beresford N. Parlett. Analysis of algorithms for reflections in bisectors. *SIAM Rev.,* 13(2):197-208, 1971.

[820] Beresford N. Parlett. *The Symmetric Eigenvalue Problem.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1980. xix+348 pp. ISBN 013-880047-2.

[821] Beresford N. Parlett. The contribution of J. H. Wilkinson to numerical analysis. In *A History of Scientific Computing,* Stephen G. Nash, editor, Addison-Wesley, Reading, MA, USA, 1990, pages 17–30.

[822] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface.* Morgan Kaufmann, San Mateo, CA, USA, 1994. xxiv+648+appendices pp. ISBN 1-55860-282-8.

[823] Vern Paxson. A program for testing IEEE decimal-binary conversion. Manuscript. `URL = ftp://ftp.ee.lbl.gov/testbase-report.ps.Z`, May 1991. 40 pp.

[824] Heinz-Otto Peitgen, Hartmut Jürgens, and Dietmar Saupe. *Fractals for the Classroom. Part one: Introduction to Fractals and Chaos.* Springer-Verlag, New York, 1992. xiv+450 pp. ISBN 0-387-97041-X.

[825] J. M. Peña. Pivoting strategies leading to small bounds of the errors for certain linear systems. *IMA J. Numer. Anal.,* 1994. Submitted.

[826] G. Peters and J. H. Wilkinson. The least squares problem and pseudo-inverses. *Comput. J.,* 13(3):309-316, 1970.

[827] G. Peters and J. H. Wilkinson. Practical problems arising in the solution of polynomial equations. *J. Inst. Maths Applics,* 8:16–35, 1971.

[828] G. Peters and J. H. Wilkinson. On the stability of Gauss-Jordan elimination with pivoting. *Comm. ACM,* 18(1):20-24, 1975.

[829] Karl Petersen. *Ergodic Theory.* Cambridge University Press, Cambridge, UK, 1981. xi+329 pp. ISBN 0-521-23632-0.

[830] M. Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numer. Math.,* 19:400–406, 1972.

[831] Daniel J. Pierce and Robert J. Plemmons. Fast adaptive condition estimation. *SIAM J. Matrix Anal. Appl.,* 13(1):274-291, 1992.

[832] Robert Piessens, Elise de Doncker-Kapenga, Christoph W. Überhuber, and David K. Kahaner. *QUADPACK: A Subroutine Package for Automatic Integration.* Springer-Verlag, Berlin, 1983. ISBN 3-540-12553-1.

[833] P. J. Plauger. Properties of floating-point arithmetic. *Computer Language,* 5(3):17-22, 1988.

[834] R. J. Plemmons. Regular splittings and the discrete Neumann problem. *Numer. Math.,* 25:153–161, 1976.

[835] Robert J. Plemmons. Linear least squares by elimination and MGS. *J. Assoc. Comput. Mach.,* 21(4):581-585, 1974.

[836] Svatopluk Poljak and Jiří Rohn. Checking robust nonsingularity is NP-hard. *Math. Control Signals Systems,* 6:1-9, 1993.

[837] Ben Polman. Incomplete blockwise factorization of (block) If-matrices. *Linear Algebra Appl.,* 90:119–132, 1987.

[838] M. J. D. Powell. A survey of numerical methods for unconstrained optimization. *SIAM Rev.,* 12(1):79-97, 1970.

[839] M. J. D. Powell. A view of unconstrained minimization algorithms that do not require derivatives. *ACM Trans. Math. Software,* 1(2):97–107, 1975.

[840] M. J. D. Powell and J. K. Reid. On applying Householder transformations to linear least squares problems. In *Pmt. IFIP Congress* 1968, North-Holland, Amsterdam, The Netherlands, 1969, pages 122-126.

[841] Stephen Power. The Cholesky decomposition in Hilbert space. *IMA Bulletin,* 22(11/12):186-187, 1986.

[842] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. Numerical *Recipes in FORTRAN: The Art of Scientific Computing.* Second edition, Cambridge University Press, Cambridge, UK, 1992. xxvi+963 pp. ISBN O 52143064 X.

[843] Douglas M. Priest. Algorithms for arbitrary precision floating point arithmetic. In *Pmt. 10th IEEE Symposium on Computer Arithmetic,* Peter Kornerup and David W. Matula, editors, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991, pages 132-143.

[844] Douglas M. Priest. *On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations.* Ph.D. thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 1992. 126 pp. URL = `ftp://ftp.icsi.berkeley.edu/pub/theory/priest-thesis.ps.Z`.

[845] J. D. Pryce. Round-off error analysis with fewer tears. *IMA Bulletin,* 17: 40-47, 1981.

[846] J. D. Pryce. A new measure of relative error for vectors. *SIAM J. Numer. Anal.,* 21(1):202-215, 1984.

[847] J. D. Pryce. Multiplicative error analysis of matrix transformation algorithms. *IMA J. Numer. Anal.,* 5:437–445, 1985.

[848] Chiara Puglisi. Modification of the Householder method based on the compact WY representation. *SIAM J. Sci. Statist. Comput.,* 13(3):723-726, 1992.

[849] Heinrich Puschmann and Joaquín Cortés. The coordinex problem and its relation to the conjecture of Wilkinson. *Numer. Math.,* 42:291–297, 1983.

[850] Heinrich Puschmann and Marcelo Nordio. Zwei Unzalässige Verstärkungen der Vermutung von Wilkinson. *Linear Algebra Appl.,* 72:167–176, 1985. In German.

[851] Gerald D. Quinlan. Round-off error in long-term orbital integrations using multistep methods. *Celestial Mechanics and Dynamical Astronomy,* 58:339–351, 1994.

[852] Kevin Quinn. Ever had problems rounding off figures? This stock exchange has. *Wall Street Journal,* 1983. 8 November.

[853] Thomas Quinn and Scott Tremaine. Roundoff error in long-term planetary orbit integrations. *Astron. J., 99(3): 1016–1023, 1990.*

[854] Thomas R. Quinn, Scott Tremaine, and Martin Duncan. A three million year integration of the earth's orbit. *Astron. J.,* 101(6):2287–2305, 1991.

[855] Ralph A. Raimi. The peculiar distribution of first digits. *Scientific American,* 221(6):109-120, 1969.

[856] Ralph A. Raimi. The first digit problem. *Amer. Math. Monthly,* 83:521-538, 1976.

[857] Louis B. Rail. *Automatic Differentiation: Techniques and Applications,* volume 120 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1981. viii+165 pp.

[858] Louis B. Rail. Tools for mathematical computation. In *Computer Aided Proofs in Analysis,* Kenneth R. Meyer and Dieter S. Schmidt, editors, volume 28 of *IMA Volumes in Mathematics and its Applications,* Springer-Verlag, New York, 1991, pages 217–228.

[859] George U. Ramos. Roundoff error analysis of the fast Fourier transform. *Math. Comp.,* 25(116):757-768, 1971.

[860] Brian Randell, editor. *The Origins of Digital Computers: Selected Papers.* Third edition, Springer-Verlag, Berlin, 1975. xvi+580 pp. ISBN 3-540- 11319-3.

[861] Wolfgang Rath. Fast Givens rotations for orthogonal similarity transformations. *Numer. Math.,* 40:47–56, 1982.

[862] Satish C. Reddy and Lloyd N. Trefethen. Stability of the method of lines. *Numer. Math.,* 62:235-267, 1992.

[863] Lothar Reichel. Newton interpolation at Leja points. *BIT,* 30:332–346, 1990.

[864] Lothar Reichel. Fast QR decomposition of Vandermond-like matrices and polynomial least squares approximation. *SIAM J. Matrix Anal. Appl.,* 12(3): 552-564, 1991.

[865] Lothar Reichel and Gerhard Opfer. Chebyshev-Vandermonde systems. *Math. Comp.,* 57(196):703-721, 1991.

[866] Lothar Reichel and Lloyd N. Trefethen. Eigenvalues and pseudo-eigenvalues of Toeplitz matrices. *Linear Algebra Appl.,* 162–164:153–185, 1992.

[867] J. K. Reid. A note on the stability of Gaussian elimination. *J. Inst. Maths Applics,* 8:374-375, 1971.

[868] J. K. Reid. Sparse matrices. In *The State of the Art in Numerical Analysis,* A. Iserles and M. J. D. Powell, editors, Oxford University Press, Oxford, UK, 1987, pages 59--85.

[869] John F. Reiser and Donald E. Knuth. Evading the drift in floating-point addition. *lnform. Process. Lett.,* 3(3):84-87, 1975.

[870] John R. Rice. Experiments on Gram-Schmidt orthogonalization. *Math. Comp.,* 20:325-328, 1966.

[871] John R. Rice. A theory of condition. *SIAM J. Numer. Anal.,* 3(2):287-310, 1966.

[872] Robert D. Richtmyer and K. W. Morton. *Difference Methods for Initial-Value Problems.* Second edition, Interscience, New York, 1967. xiv+405 pp.

[873] J. L. Rigal and J. Gaches. On the compatibility of a given solution with the data of a linear system. *J. Assoc. Comput. Mach.,* 14(3):543-548, 1967.

[874] T. G. Robertazzi and S. C. Schwartz. Best "ordering" for floating-point addition. *ACM Trans. Math. Software,* 14(1):101–110, 1988.

[875] J. D. Roberts. Linear model reduction and solution of the algebraic Riccati equation by use of the sign function. *Int. J. Control,* 32(4):677-687, 1980. First issued as report CUED/B-Control/TR13, Department of Engineering, University of Cambridge, 1971.

[876] Jiří Rohn. New condition numbers for matrices and linear systems. *Computing,* 41:167–169, 1989.

[877] Jiří Rohn. Systems of linear interval equations. *Linear Algebra Appl.,* 126: 39-78, 1989.

[878] Jiří Rohn. Nonsingularity under data rounding. *Linear Algebra Appl.,* 139: 171–174, 1990.

[879] Jiří Rohn. NP-hardness results for some linear and quadratic problems. Technical Report No. 619, Institute of Computer Science, Academy of Sciences of the Czech Reupublic, Prague, January 1995. 11 pp.

[880] D. R. Ross. Reducing truncation errors using cascading accumulators. *Comm. ACM,* 8(1):32-33, 1965.

[881] M. W. Routh, P. A. Swartz, and M. B. Denton. Performance of the super modified simplex. *Analytical Chemistry,* 49(9):1422–1428, 1977.

[882] Thomas Harvey Rowan. *Functional Stability Analysis of Numerical Algorithms.* Ph.D. thesis, University of Texas at Austin, Austin, TX, USA, May 1990. xii+205 pp.

[883] Axel Ruhe. Numerical aspects of Gram-Schmidt orthogonalization of vectors. *Linear Algebra Appl.,* 52/53:591-601, 1983.

[884] D. E. Rutherford. Some continuant determinants arising in physics and chemistry. *Pmt. Royal Sot. Edinburgh,* 62,A:229–236, 1947.

[885] D. E. Rutherford. Some continuant determinants arising in physics and chemistry—II. *Proc. Royal Soc. Edinburgh,* 63,A:232-241, 1952,

[886] H. Rutishauser. On test matrices. In *Programmation en Mathématiques Numériques, Besançon,* 1966, volume 7 (no. 165) of *Éditions Centre Nat. Recherche Sci., Paris,* 1968, pages 349-365.

[887] Heinz Rutishauser. Solution of eigenvalue problems with the LR-transformation. In *Further Contributions to the Solution of Simultaneous Linear Equations and the Determination of Eigenvalues,* number 49 in Applied Mathematics Series, National Bureau of Standards, United States Department of Commerce, Washington, DC, 1958, pages 47–81.

[888] *FTN90 Reference Manual.* Second edition, Salford Software Ltd. and the Numerical Algorithms Group Ltd., Salford and Oxford, UK, 1993.

[889] Ahmed H. Sameh and Richard P. Brent. Solving triangular systems on a parallel computer. *SIAM J. Numer. Anal.,* 14(6):1101–1113, 1977.

[890] Klaus Samelson and Friedrich L. Bauer. Optimale Rechengenauigkeit bei Rechenanlagen mit gleitendem Komma. *Zeitschrift für Angewandte Mathematik und Physik,* 4:312–316, 1953.

[891] J. M. Sanz-Serna. Symplectic integrators for Hamiltonian problems: An overview. In *Acta Numerica,* Cambridge University Press, Cambridge, UK, 1992, pages 243–286.

[892] J. M. Sanz-Serna and M. P. Calve. Numerical *Hamiltonian Problems.* Chap man and Hall, London, 1994. xii+207 pp. ISBN 0-412-54290-0.

[893] J. M. Sanz-Serna and S. Larsson. Shadows, chaos, and saddles. *Appl. Numer. Math.,* 13:181-190, 1993.

[894] M. A. Saunders. Large-scale linear programming using the Cholesky factorization. Report CS 252, Department of Computer Science, Stanford University, January 1972.

[895] Werner Sautter. Error analysis of Gauss elimination for the best least squares problem. *Numer. Math.,* 30:165–184, 1978.

[896] I. Richard Savage and Eugene Lukacs. Tables of inverse of finite segments of the Hilbert matrix. In *Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues$_j$* Olga Taussky, editor, number 39 in Applied Mathematics Series, National Bureau of Standards, United States Department of Commerce, Washington, DC, 1954, pages 105-108.

[897] James B. Scarborough. Numerical *Mathematical Analysis.* Second edition, Johns Hopkins University Press, Baltimore, MD, USA, 1950. xviii+511 pp.

[898] Charles W. Schelin. Calculator function approximation. *Amer. Math. Monthly,* 90(5):317-325, 1983.

[899] R. Scherer and K. Zeller. Shorthand notation for rounding errors. *Computing*, Suppl. 2:165-168, 1980.

[900] Hans Schneider and W. Gilbert Strang. Comparison theorems for supremum norms. *Numer. Math.,* 4:15–20, 1962.

[901] J. L. Schonfelder and M. Razaz. Error control with polynomial approximations. *IMA J. Numer. Anal.,* 1:105-114, 1980.

[902] Robert S. Schreiber. Block algorithms for parallel machines. In *Numerical Algorithms for Modern Parallel Computer Architectures,* M. H. Schultz, editor, number 13 in IMA Volumes In Mathematics and Its Applications, Springer-Verlag, Berlin, 1988, pages 197-207.

[903] Robert S. Schreiber. Hough's random story explained. *NA Digest,* Volume 89, Issue 3, 1989. Electronic mail magazine: `na.help@na-net.ornl.gov.`

[904] Robert S. Schreiber and Beresford N. Parlett. Block reflectors: Theory and computation. *SIAM J. Numer. Anal.,* 25(1):189-205, 1988.

[905] Robert S. Schreiber and Charles F. Van Loan. A storage efficient WY representation for products of Householder transformations. *SIAM J. Sci. Statist. Comput.,* 10:53-57, 1989.

[906] N. L. Schryer. A test of a computer's floating-point arithmetic unit. Computing Science Technical Report No. 89, AT&T Bell Laboratories, Murray Hill, NJ, USA, 1981.

[907] N. L. Schryer. Determination of correct floating-point model parameters. In *Sources and Development of Mathematical Software,* Wayne R. Cowell, editor, Prentice-Hall, Englewood Cliffs, NJ, USA, 1984, pages 360-366.

[908] Gunther Schulz. Iterative Berechnung der reziproken Matrix. Z. *Angew. Math. Mech.,* 13:57-59, 1933.

[909] Robert Sedgewick. *Algorithms*. Second edition, Addison-Wesley, Reading, MA, USA, 1988. xii+657 pp. ISBN 0-201-06673-4.

[910] Lawrence F. Sharnpine. Numerical *Solution of Ordinary Differential Equations.* Chapman and Hall, New York$_j$ 1994. x+484 pp. ISBN 0-412-05151-6.

[911] Lawrence F. Sharnpine and Richard C. Allen, Jr. *Numerical Computing: An Introduction*. W. B. Saunders, Philadelphia, PA, USA, 1973. vii+258 pp. ISBN 0-7216-8150-6.

[912] Lawrence F. Shampine and Mark W. Reichelt. The MATLAB ODE suite. Manuscript, 1995. 35 pp.

[913] Alexander Shapiro. Optimally scaled matrices, necessary and sufficient conditions. *Numer. Math.,* 39:239-245, 1982.

[914] Alexander Shapiro. Optimal block diagonal l_2-scaling of matrices. *SIAM J. Numer. Anal.,* 22(1):81-94, 1985.

[915] Alexander Shapiro. Upper bounds for nearly optimal diagonal scaling of matrices. *Linear and Multilineal Algebra,* 29:147–147, 1991.

[916] H. P. Sharangpani and M. L. Barton. Statistical analysis of floating point flaw in the Pentium processor (1994). Technical report, Intel Corporation, November 1994. 31 pp.

[917] David Shepherd and Greg Wilson. Making chips that work. New *Scientist,* pages 61–64, 1989. 13 May.

[918] Gautam M. Shroff and Christian H. Bischof. Adaptive condition estimation for rank-one updates of QR factorization. *SIAM J. Matrix Anal. Appl.,* 13 (4):1264-1278, 1992.

[919] Robert D. Skeel. Scaling for numerical stability in Gaussian elimination. *J. Assoc. Comput. Mach.,* 26(3):494-526, 1979.

[920] Robert D. Skeel. Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comp.,* 35(151):817–832, 1980.

[921] Robert D. Skeel. Effect of equilibration on residual size for partial pivoting. *SIAM J. Numer. Anal.,* 18(3):449-454, 1981.

[922] Robert D. Skeel. Safety in numbers: The boundless errors of numerical computation. Working Document 89-3, Department of Computer Science, University of Illinois, Urbana, IL, USA, 1989. 9 pp.

[923] Robert D. Skeel and Jerry B. Keiper. *Elementary Numerical Computing with Mathematical.* McGraw-Hill, New York, 1993. xiv+434 pp. ISBN 0-07-057820-6.

[924] Steve Smale. Some remarks on the foundations of numerical analysis. *SIAM Rev.,* 32(2):211-220, 1990.

[925] B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler. *Matrix Eigensystem Routines—EISPACK Guide,* volume 6 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, 1976.

[926] David M. Smith. Algorithm 693: A FORTRAN package for floating-point multiple-precision arithmetic. *ACM Trans. Math. Software,* 17(2):273-283, 1991.

[927] Francis J. Smith. An algorithm for summing orthogonal polynomial series and their derivatives with applications to curve-fitting and interpolation. *Math. Comp.,* 19:33-36, 1965.

[928] Jon M. Smith. *Scientific Analysis on the Pocket Calculator.* Wiley, New York, 1975. xii+380 pp. ISBN 0-471-79997-1.

[929] Robert L. Smith. Algorithm 116: Complex division. *Comm. ACM,* 5(8):435, 1962.

[930] Alicja Smoktunowicz. A note on the strong componentwise stability of algorithms for solving symmetric linear systems. *Demonstratio Mathematical,* 28 (2), 1995.

[931] Alicja Smoktunowicz and Jolanta Sokolnicka. Binary cascades iterative refinement in doubled-mantissa arithmetics. *BIT,* 24:123–127, 1984.

[932] James N. Snyder. On the improvement of the solutions to a set of simultaneous linear equations using the ILLIAC. *Mathematical Tables and Other Aids to Computation,* 9:177–184, 1955.

[933] P. Spellucci. An approach to backward analysis for linear and nonlinear iterative methods. *Computing,* 25:269–282, 1980.

[934] J. Spieiß. Untersuchungen des Zeitgewinns durch neue Algorithmen zur Matrix-Multiplikation. *Computing,* 17:23-36, 1976.

[935] Gerhard Starke and Wilhelm Niethammer. SOR for $AX - XB = C$. *Linear Algebra Appl.,* 154-156:355-375, 1991.

[936] Guy L. Steele, Jr. and Jon L. White. How to print floating-point numbers accurately. *SIGPLAN Notices,* 25(6):112–126, 1990.

[937] Irene A. Stegun and Milton Abrarnowitz. Pitfalls in computation. *J. Soc. Indust. Appl. Math.,* 4(4):207-219, 1956.

[938] Pat H. Sterbenz. *Floating-Point Computation.* Prentice-Hall, Englewood Cliffs, NJ, USA, 1974. xiv+316 pp. ISBN 0-13-322495-3.

[939] G. W. Stewart. Error analysis of the algorithm for shifting the zeros of a polynomial by synthetic division. *Math. Comp.,* 25(113):135-139, 1971.

[940] G. W. Stewart. Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Rev.,* 15(4):727-764, 1973.

[941] G. W. Stewart. *Introduction to Matrix Computations.* Academic Press, New York, 1973. xiii+441 pp. ISBN 0-12-670350-7.

[942] G. W. Stewart. Modifying pivot elements in Gaussian elimination. *Math. Comp.,* 28(126):537-542, 1974.

[943] G. W. Stewart. On the perturbation of pseudo-inverses, projections and linear least squares problems. *SIAM Rev.,* 19(4):634-662, 1977.

[944] G. W. Stewart. Perturbation bounds for the *QR* factorization of a matrix. *SIAM J. Numer. Anal.,* 14(3):509-518, 1977.

[945] G. W. Stewart. Research, development, and LINPACK. In *Mathematical Software III,* John R. Rice, editor, Academic Press, New York, 1977, pages 1–14.

[946] G. W. Stewart. The efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.,* 17(3):403–409, 1980.

[947] G. W. Stewart. A note on complex division. *ACM Trans. Math. Software,* 11(3):238-241, 1985.

[948] G. W. Stewart. Stochastic perturbation theory. *SIAM Rev.,* 32(4):579+10, 1990.

[949] G. W. Stewart. Note on a generalized Sylvester equation. IMA Preprint Series #985, Institute for Mathematics and its Applications, University of Minnesota, Minneapolis, MN, USA, May 1992.3 pp.

[950] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Rev.,* 35(4):551-566, 1993.

[951] G. W. Stewart. On the perturbation of LU, Cholesky, and QR factorizations. *SIAM J. Matrix Anal. Appl.,* 4(4):1141-1145, 1993.

[952] G. W. Stewart. Gauss, statistics, and Gaussian elimination. Technical Report TR-3307, Department of Computer Science, University of Maryland, College Park, MD, USA, August 1994. 14 pp.

[953] G. W. Stewart. On Markov chains with sluggish transients. Technical Report TR-3306, Department of Computer Science, University of Maryland, College Park, MD, USA, June 1994. 8 pp.

[954] G. W. Stewart and Ji-guang Sun. *Matrix Perturbation Theory.* Academic Press, London, 1990. xv+365 pp. ISBN 0-12-670230-6.

[955] J. Steer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, 1980. ix+609 pp. ISBN 0-387-90420-4.

[956] J. Steer and C. Witzgall. Transformations by diagonal matrices in a normed space. *Numer. Math.,* 4:158–171, 1962.

[957] Betty Jane Stone. Best possible ratios of certain matrix norms. *Numer. Math.,* 4:114-116, 1962.

[958] David R. Stoutemyer. Automatic error analysis using computer algebraic manipulation. *ACM Trans. Math. Software,* 3(1):26–43, 1977.

[959] David R. Stoutemyer. Crimes and misdemeanours in the computer algebra trade. *Notices Amer. Math. Soc.,* 38(7):778–785, 1991.

[960] Gilbert Strang. A proposal for Toeplitz matrix calculations. *Stud. Appl. Math.,* 74:171-176, 1986.

[961] Gilbert Strang. *Introduction to Linear Algebra.* Wellesley-Cambridge Press, Wellesley, MA, USA, 1993. viii+472 pp. ISBN 0-9614088-5-5.

[962] V. Strassen. Gaussian elimination is not optimal. *Numer. Math.,* 13:354-356, 1969.

[963] F. Stummel. Rounding error analysis of elementary numerical algorithms. *Computing,* Suppl. 2:169-195, 1980.

[964] F. Stummel. Perturbation theory for evaluation algorithms of arithmetic expressions. *Math. Comp.,* 37(156):435–473, 1981.

[965] F. Stummel. Optimal error estimates for Gaussian elimination in floating-point arithmetic. Z. *Angew. Math. Mech.,* 62:T355–T357, 1982.

[966] F. Stummel. Strict optimal error estimates for Gaussian elimination. Z. *Angew. Math. Mech.,* 65:T405-T407, 1985.

[967] Friedrich Stummel. Forward error analysis of Gaussian elimination, Part I: Error and residual estimates. *Numer. Math.,* 46:365-395, 1985.

[968] Friedrich Stummel. Forward error analysis of Gaussian elimination, Part II: Stability theorems. *Numer. Math.,* 46:397-415, 1985.

[969] *SPARCompiler FORTRAN* 2.0.1: *User's Guide*. Sun Microsystems, Inc., Mountain View, CA, USA, October 1992. Part No. 800-6552-11, Revision A.

[970] *SPARCompiier FORTRAN: Numerical Computation Guide*. Sun Microsystems, Inc., Mountain View, CA, USA, October 1992. Part No. 800-7097-11, Revision A.

[971] Ji-guang Sun. Perturbation bounds for the Cholesky and *QR* factorization. *BIT*, 31:341-352, 1991.

[972] Ji-guang Sun. Componentwise perturbation bounds for some matrix decompositions. *BIT*, 32(4):702–714, 1992.

[973] Ji-guang Sun. Rounding-error and perturbation bounds for the Cholesky and LDL^T factorization. *Linear Algebra Appl.*, 173:77-97, 1992.

[974] Ji-guang Sun. A note on backward perturbations for the Hermitian eigenvalue problem. *BIT*, 35, 1995. To appear.

[975] Ji-guang Sun. On perturbation bounds for the QR factorization. *Linear Algebra Appl.*, 215:95-111, 1995.

[976] Ji-guang Sun. Optimal backward perturbation bounds for the linear LS problem with multiple right-hand sides. *IMA J. Numer. Anal.*, 1996. To appear.

[977] Xiaobai Sun and Christian H. Bischof. A basis-kernel representation of orthogonal matrices. *SIAM J. Matrix Anal. Appl.*, 16(4):1184-1196, 1995.

[978] W. H. Swarm. Direct search methods. In Numerical Methods for *Unconstrained Optimization*, W. Murray, editor, Academic Press, London, 1972, pages 13-28.

[979] W. H. Swarm. Constrained optimization by direct search. In *Numerical Methods for Constrained Optimization*, P. E. Gill and W. Murray, editors, Academic Press, London, 1974, pages 191-217.

[980] Earl E. Swartzlander, Jr., editor. *Computer Arithmetic*, volume 21 of *Benchmark Papers in Electrical Engineering and Computer Science*. Dowden, Hutchinson and Ross, Stroudsburg, PA, USA, 1980.

[981] Earl E. Swartzlander, Jr. and Aristides G. Alexopolous. The sign/logarithm number system. *IEEE Trans. Comput.*, C-24(12):1238–1242, 1975.

[982] D. W. Sweeney. An analysis of floating-point addition. *IBM Systems Journal*, 4:31-42, 1965. Reprinted in [980, pp. 317-328].

[983] J. J. Sylvester. Additions to the articles, "On a New Class of Theorems," and "On Pascal's Theorem". *Philosophical Magazine*, 37:363-370, 1850. Reprinted in [986, pp. 1451-151].

[984] J. J. Sylvester. On the relation between the minor determinants of linearly equivalent quadratic functions. *Philosophical Magazine*, (Fourth Series) 1: 295-305, 1851. Reprinted in [986, pp. 241-250].

[985] J. J. Sylvester. Sur l'equation en matrices *px* = *xq*. *Comptes Rendus de l'Académie des Sciences*, pages 67-71 and 115-116, 1884.

[986] *The Collected Mathematical Papers of James Joseph Sylvester,* volume 1 (1837-1853). Cambridge University Press, Cambridge, UK,1904. xii+650 pp.

[987] Ping Tak Peter Tang. Table-driven implementation of the exponential function in IEEE floating-point arithmetic. *ACM Trans. Math. Software,* 15(2): 144–157, 1989.

[988] Ping Tak Peter Tang. Accurate and efficient testing of the exponential and logarithm functions. *ACM Trans. Math. Software,* 16(3):185-200, 1990.

[989] Ping Tak Peter Tang. Table-driven implementation of the logarithm function in IEEE floating-point arithmetic. *ACM Trans. Math. Software*, 16(3):378–400, 1990.

[990] Ping Tak Peter Tang. Table-lookup algorithms for elementary functions and their error analysis. In *Proc. 10th IEEE Symposium on Computer Arithmetic,* Peter Kornerup and David W. Matula, editors, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991, pages 232–236.

[991] Ping Tak Peter Tang. Table-driven implementation of the expm1 function in IEEE floating-point arithmetic. *ACM Trans. Math. Software*, 18(2):211-222, 1992.

[992] W. P. Tang and G. H. Golub. The block decomposition of a Vandermonde matrix and its applications. *BIT,* 21:505–517, 1981.

[993] Pham Dinh Tao. Convergence of a subgradient method for computing the bound norm of matrices. *Linear Algebra Appl.,* 62:163–182, 1984. In French.

[994] Pham Dinh Tao. Some methods for computing the maximum of quadratic form on the unit ball of the maximum norm. *Numer. Math.,* 45:377–401, 1984. In French.

[995] A. H. Taub, editor. *John von Neumann Collected Works,* volume V, Design of Computers, Theory of Automata and Numerical Analysis. Pergamon, Oxford, UK, 1963. ix+784 pp.

[996] Olga Taussky. A remark concerning the characteristic roots of the finite segments of the Hilbert matrix. Quart. *J. Math.,* 20:80–83, 1949.

[997] Olga Taussky. How I became a torchbearer for matrix theory. *Amer. Math. Monthly,* 95(9):801–812, November 1988.

[998] Olga Taussky and Marvin Marcus. Eigenvalues of finite matrices. In Survey of *Numerical Analysis,* John Todd, editor, McGraw-Hill, New York, 1962, pages 279–313.

[999] Henry C. Thacher, Jr. Algorithm 43: Crout with pivoting II. *Comm. ACM,* 4(4):176-177, 1961.

[1000] Ronald A. Thisted. *Elements of Statistical Computing: Numerical Computation.* Chapman and Hall, New York, 1988. xx+427 pp. ISBN 0-412-01371-1.

[1001] D'Arcy Wentworth Thompson. On *Growth and Form. The Complete Revised Edition*. Cambridge University Press, 1942. viii+1116 pp. Reprinted by Dover, New York, 1992. ISBN 0-486-67135-6.

[1002] Martti Tienari. A statistical model of roundoff error for varying length floating-point arithmetic. *BIT,* 10:355-365, 1970.

[1003] J. Todd. On condition numbers. In *Programmation en Mathématiques Numérques, Besançon, 1966,* volume 7 (no. 165) of *Éditions Centre Nat. Recherche Sci., Paris,* 1968, pages 141-159.

[1004] John Todd. The condition of the finite segments of the Hilbert matrix. In *Contributions to the Solution of Systems of Linear Equations and the Determination of Eigenvalues,* Olga Taussky, editor, number 39 in Applied Mathematics Series, National Bureau of Standards, United States Department of Commerce, Washington, DC, 1954, pages 109-116.

[1005] John Todd. Computational problems concerning the Hilbert matrix. *J. Res. National Bureau Standards-B,* 65(1):19-22, 1961.

[1006] John Todd. *Basic Numerical Mathematics, Vol.* 2: *Numerical Algebra.* Birkhäuser, Basel, and Academic Press, New York, 1977. 216 pp. ISBN 0-12-692402-3.

[1007] Kim-Chuan Toh and Lloyd N. Trefethen. Pseudozeros of polynomials and pseudospectra of companion matrices. *Numer. Math.,* 68(3):403-425, 1994.

[1008] Virginia J. Torczon. *Multi-Directional Search: A Direct Search Algorithm for Parallel Machines.* Ph.D. thesis, Rice University, Houston, TX, USA, May 1989. vii+85 pp.

[1009] Virginia J. Torczon. On the convergence of the multidirectional search algorithm. *SIAM J. Optim.,* 1(1):123–145, 1991.

[1010] Virginia J. Torczon. PDS: Direct search methods for unconstrained optimization on either sequential or parallel machines. Report TR92-9, Department of Mathematical Sciences, Rice University, Houston, TX, USA, March 1992. To appear in ACM Trans. Math. Software.

[1011] Virginia J. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.,* 7(1), 1997. To appear.

[1012] L. Tornheim. Maximum third pivot for Gaussian reduction. Technical report, Calif. Research Corp., Richmond, CA, USA, 1965. Cited in [256].

[1013] J. F. Traub. Associated polynomials and uniform methods for the solution of linear problems. *SIAM Rev.,* 8(3):277-301, 1966.

[1014] Lloyd N. Trefethen. Three mysteries of Gaussian elimination. *ACM SIGNUM Newsletter,* 20:2-5, 1985.

[1015] Lloyd N. Trefethen. Approximation theory and numerical linear algebra. In *Algorithms for Approximation* II, J. C Mason and M. G. Cox, editors, Chapman and Hall, London, 1990, pages 336-360.

[1016] Lloyd N. Trefethen. The definition of numerical analysis. *SIAM News,* 25: 6 and 22, November 1992. Reprinted in *IMA Bulletin,* 29 (3/4), pp. 47-49, 1993.

[1017] Lloyd N. Trefethen. Pseudospectra of matrices. In *Numerical Analysis* 1991, *Proceedings of the 14th Dundee Conference,* D. F. Griffiths and G. A. Watson, editors, volume 260 of *Pitman Research Notes in Mathematics,* Longman Scientific and Technical, Essex, UK, 1992, pages 234-266.

[1018] Lloyd N. Trefethen. *Spectra and Pseudospedra: The Behavior of Non-Normal Matrices and Operators*. Book in preparation.

[1019] Lloyd N. Trefethen and Robert S. Schreiber. Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl.,* 11(3):335–360, 1990.

[1020] Lloyd N. Trefethen and Manfred R. Trummer. An instability phenomenon in spectral methods. *SIAM J. Numer. Anal.,* 24(5):1008–1023, 1987.

[1021] Henry S. Tropp. FORTRAN anecdotes. *Ann. Hist. Comput.,* 6(1):59-64, 1984.

[1022] Henry S. Tropp. Origin of the term *bit*. *Ann. Hist. Comput.,* 6(2):152-155, 1984.

[1023] Nai-kuan Tsao. A note on implementing the Householder transformation. *SIAM J. Numer. Anal.,* 12(1):53-58, 1975.

[1024] Nai-kuan Tsao. A simple approach to the error analysis of division-free numerical algorithms. *IEEE Trans. Comput.,* C-32(4):343–351, 1983.

[1025] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. London Math. Soc.,* 42:230-265, 1936.

[1026] A. M. Turing. Proposal for development in the Mathematics Division of an Automatic Computing Engine (ACE). Report E.882, Executive Committee, National Physical Laboratory, Teddington, UK, 1945. Reprinted in [186, pp. 20-105] and [603, pp. 1-86].

[1027] A. M. Turing. Rounding-off errors in matrix processes. Quart. *J. Mech. Appl. Math.,* 1:287–308, 1948. Reprinted in [149] with summary and notes (including corrections).

[1028] H. W. Turnbull. *The Theory of Determinants, Matrices, and Invariant*. Blackie, London and Glasgow, 1929. xvi+338 pp.

[1029] H. W. Turnbull and A. C. Aitken. *An Introduction to the Theory of Canonical Matrices*. Blackie, London and Glasgow, 1932. xiii+200 pp. Reprinted with appendix, 1952.

[1030] Kathryn Turner. Computing projections for the Karmakar algorithm. *Linear Algebra Appl.,* 152:141-154, 1991.

[1031] Peter R. Turner. The distribution of leading significant digits. *IMA J. Numer. Anal.,* 2:407-412, 1982.

[1032] Peter R. Turner. Further revelations on L.S.D. *IMA J. Numer. Anal.,* 4: 225-231, 1984.

[1033] Peter R. Turner. Will the “real” real arithmetic please stand up? *Notices Amer. Math. Soc.,* 38(4):298-304, 1991.

[1034] Evgenij E. Tyrtyshnikov. Cauchy-Toeplitz matrices and some applications. *Linear Algebra Appl.,* 149:1-18, 1991.

[1035] Evgenij E. Tyrtyshnikov. How bad are Hankel matrices? *Numer. Math.*, 67 (2):261-269, 1994.

[1036] Patriot missile defense: Software problem led to system failure at Dhahran, Saudi Arabia. Report GAO/IMTEC-92-26, Information Management and Technology Division, United States General Accounting Office, Washington, DC, February 1992. 16 pp.

[1037] Minoru Urabe. Roundoff error distribution in fixed-point multiplication and a remark about the rounding rule. *SIAM J. Numer. Anal.*, 5(2):202-210, 1968.

[1038] J. V. Uspensky. *Theory of Equations.* McGraw-Hill, New York, 1948. vii+353 pp.

[1039] A. van der Sluis. Condition numbers and equilibration of matrices. *Numer. Math.*, 14:14-23, 1969.

[1040] A. van der Sluis. Condition, equilibration and pivoting in linear algebraic systems. *Numer. Math.*, 15:74–86, 1970.

[1041] A. van der Sluis. Stability of the solutions of linear least squares problems. *Numer. Math.*, 23:241-254, 1975.

[1042] Charles F. Van Loan. On the method of weighting for equality-constrained least-squares problems. *SIAM J. Numer. Anal.*, 22(5):851-864, 1985.

[1043] Charles F. Van Loan. On estimating the condition of eigenvalues and eigenvectors. *Linear Algebra Appl.*, 88/89:715–732, 1987.

[1044] Charles F. Van Loan. *Computational Frameworks for the Fast Fourier Transform*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992. xiii+273 pp. ISBN 0-89871-285-8.

[1045] M. van Veldhuizen. A note on partial pivoting and Gaussian elimination. *Numer. Math.*, 29:1-10, 1977.

[1046] A. van Wijngaarden. Numerical analysis as an independent science. *BIT*, 6: 66-81, 1966.

[1047] Robert J. Vanderbei. Symmetric quasi-definite matrices. *SIAM J. Optim.*, 5 (1):100-113, 1995.

[1048] J. M. Varah. On the solution of block-tridiagonal systems arising from certain finit-difference equations. *Math. Comp.*, 26(120):859-868, 1972.

[1049] J. M. Varah. A lower bound for the smallest singular value of a matrix. *Linear Algebra Appl.*, 11:3-5, 1975.

[1050] J. M. Varah. On the separation of two matrices. *SIAM J. Numer. Anal.*, 16 (2):216-222, 1979.

[1051] Richard S. Varga. On diagonal dominance arguments for bounding $\|A^{-1}\|_\infty$. *Linear Algebra Appl.*, 14:211-217, 1976.

[1052] Richard S. Varga. *Scientific Computation on Mathematical Problems and Conjectures*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1990. vi+122 pp. ISBN 0-89871-257-2.

[1053] Rank M. Verzuh. The solution of simultaneous linear equations with the aid of the 602 calculating punch. *M.T.A.C.,* 3:453–462, 1949.

[1054] J. Vignes and R. Alt. An efficient stochastic method for round-off error analysis. In *Accurate Scientific Computations, Proceedings,* 1985, Willard L. Miranker and Richard A. Toupin, editors, volume 235 of *Lecture Notes in Computer Science,* Springer-Verlag, Berlin, 1986, pages 183-205.

[1055] Emil Vitasek. The numerical stability in solution of differential equations. In *Conference on the Numerical Solution of Differential Equations,* J. L1. Morris, editor, volume 109 of *Lecture Notes in Mathematics,* Springer-Verlag, Berlin, 1969, pages 87–111.

[1056] I. V. Viten'ko. Optimum algorithms for adding and multiplying on computers with a floating point. *U.S.S.R. Comput. Math. Math. Phys.,* 8(5):183-195, 1968.

[1057] John von Neumann and Herman H. Goldstine. Numerical inverting of matrices of high order. *Bull. Amer. Math. Soc.,* 53:1021–1099, 1947. Reprinted in [995, pp. 479–557].

[1058] H. A. Van Der Vorst. The convergence behaviour of preconditioned CG and CG-S in the presence of rounding errors. In *Preconditioned Conjugate Gradient Methods,* Owe Axelsson and Lily Yu. Kolotilina, editors, volume 1457 of *Lecture Notes in Mathematics,* Springer-Verlag, Berlin, 1990, pages 126–136.

[1059] Eugene L. Wachpress. Iterative solution of the Lyapunov matrix equation. *Appl. Math. Lett.,* 1(1):87-90, 1988.

[1060] Bertil Waldén, Rune Karlson, and Ji-guang Sun. Optimal backward perturbation bounds for the linear least squares problem. *Numerical Linear Algebra with Applications,* 2(3):271–286, 1995.

[1061] Peter J. L. Wallis, editor. *Improving Floating-Point Programming.* Wiley, London, 1990. xvi+191 pp. ISBN 0-471-92437-7.

[1062] W. D. Wallis. Hadarnard matrices. In *Combinatorial and Graph- Theoretical Problems in Linear Algebra,* Richard A. Brualdi, Shmuel Friedland, and Victor Klee, editors, volume *50* of *IMA Volumes in Mathematics and its Applications,* Springer-Verlag, New York, 1993, pages 235–243.

[1063] W. D. Wallis, Anne Penfold Street$_j$ and Jennifer Seberry Wallis. *Combinatorics: Room Squares, Sum-Free Sets, Hadamard Matrices,* volume 292 of Lecture *Notes in Mathematics.* Springer-Verlag, Berlin, 1972. 508 pp. ISBN 3-540-06035-9.

[1064] Johnson J. H. Wang. *Generalized Moment Methods in Electromagnetic: Formulation and Computer Solution of Integral Equations.* Wiley, New York, 1991. xiii+553 pp. ISBN 0-471-51443-8.

[1065] Robert C. Ward. The QR algorithm and Hyman's method on vector computers. *Math. Comp.,* 30(133):132–142, 1976.

[1066] Willis H. Ware, editor. Soviet computer technology–1959. *Comm. ACM,* 3 (3):131-166, 1960.

[1067] G. A. Watson. An algorithm for optimal l_2 scaling of matrices. *IMA J. Numer. Anal.,* 11:481-492, 1991.

[1068] J. H. M. Wedderburn. *Lectures On Matrices,* volume 17 of *American Mathematical Society Colloquium Publications.* American Mathematical Society, Providence, RI, USA, 1934. vii+205 pp.

[1069] Per-Åke Wedin. Perturbation theory for pseudo-inverses. *BIT,* 13:217-232, 1973.

[1070] Per-Åke Wedin. Perturbation theory and condition numbers for generalized and constrained linear least squares problems. Report UMINF 125.85, Institute of Information Processing, University of Umeå, Umeå, Sweden, May 1985.

[1071] Elias Wegert and Lloyd N. Trefethen. From the Buffon needle problem to the Kreiss matrix theorem. *Amer. Math. Monthly,* 101(2):132–139, 1994.

[1072] Musheng Wei. Perturbation of the least squares problem. *Linear Algebra Appl.,* 141:177-182, 1990.

[1073] N. Weiss, G. W. Wasilkowski, H. Woźniakowski, and M. Shub. Average condition number for solving linear equations. *Linear Algebra Appl.,* 83: 79-102, 1986.

[1074] Burton Wendroff. *Theoretical Numerical Analysis.* Academic Press, New York, 1966. xi+239 pp.

[1075] Wilhelm Werner. Polynomial interpolation: Lagrange versus Newton. *Math. Comp.,* 43(167):205-217, 1984.

[1076] Joan R. Westlake. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations.* Wiley, New York, 1968.

[1077] B. A. Wichmann. Towards a formal specification of floating point. *Comput. J.,* 32:432-436, 1989.

[1078] B. A. Wichmann. A note on the use of floating point in critical systems. *Comput. J.,* 35(1):41-44, 1992.

[1079] J. H. Wilkinson. The Automatic Computing Engine at the National Physical Laboratory. *Proc. Roy. Sot. London Ser. A,* 195:285-286, 1948.

[1080] J. H. Wilkinson. Progress report on the Automatic Computing Engine. Report MA/17/1024, Mathematics Division, Department of Scientific and Industrial Research, National Physical Laboratory, Teddington, UK, April 1948. 127 pp.

[1081] J. H. Wilkinson. Linear algebra on the Pilot ACE. In *Automatic Digital Computation,* Her Majesty's Stationery Office, London, 1954. Reprinted in [1104, pp. 337-344].

[1082] J. H. Wilkinson. The Pilot ACE. In *Automatic Digital Computation,* Her Majesty's Stationery Office, London, 1954, pages 5-14. Reprinted in [87, pp. 193-199] and [1104, pp. 219-228].

[1083] J, H. Wilkinson. The use of iterative methods for finding the latent roots and vectors of matrices. *Mathematical Tables and Other Aids to Computation,* 9: 184–191, 1955.

[1084] J. H. Wilkinson. Error analysis of floating-point computation. *Numer. Math.,* 2:319-340, 1960.

[1085] J. H. Wilkinson. Error analysis of direct methods of matrix inversion. *J. Assoc. Comput. Mach.,* 8:281–330, 1961.

[1086] J. H. Wilkinson. Error analysis of eigenvalue techniques based on orthogonal transformations. *J. Sot. Indust. Appl. Math.,* 10(1):162–195, 1962.

[1087] J. H. Wilkinson. Plan-rotations in floating-point arithmetic. In *Experimental Arithmetic, High Speed Computing and Mathematics,* volume 15 of *Proceedings of Symposia in Applied Mathematics,* American Mathematical Society, Providence, RI, USA, 1963, pages 185-198.

[1088] J. H. Wilkinson. *Rounding Errors in Algebraic Processes.* Notes on Applied Science No. 32, Her Majesty's Stationery Office, London, 1963. vi+161 pp. Also published by Prentice-Hall, Englewood Cliffs, NJ, USA. Reprinted by Dover, New York, 1994. ISBN 0-486-67999-3.

[1089] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Oxford University Press, 1965. xviii+662 pp. ISBN 0-19-853403-5 (hardback), 0-19-853418-3 (paperback) .

[1090] J. H. Wilkinson. Error analysis of transformations based on the use of matrices of the form $I - 2ww^H$. In *Error In Digital Computation,* Louis B. Rail, editor, volume 2, Wiley, New York, 1965, pages 77–101.

[1091] J. H. Wilkinson. *Bledy Zaokragleń w Procesach Algebraicznych.* PWW, Warszawa, 1967. Polish translation of [1088].

[1092] J. H. Wilkinson. A priori error analysis of algebraic processes. In *Proc. International Congress of Mathematicians, Moscow 1966,* I. G. Petrovsky, editor, Mir Publishers, Moscow, 1968, pages 629–640.

[1093] J. H. Wilkinson. *Rundungsfehler.* Springer-Verlag, Berlin, 1969. German translation of [1088].

[1094] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Nauka, U.S.S.R. Academy of Sciences, 1970.564 pp. Russian translation of [1089].

[1095] J. H. Wilkinson. Modern error analysis. *SIAM Rev.,* 13(4):548-568, 1971.

[1096] J. H. Wilkinson. Some comments from a numerical analyst. *J. Assoc. Comput. Mach.,* 18(2):137-147, 1971. Reprinted in [3].

[1097] J. H. Wilkinson. The classical error analyses for the solution of linear systems. *IMA Bulletin,* 10(5/6):175-180, 1974.

[1098] J. H. Wilkinson. Numerical linear algebra on digital computers. *IMA Bulletin,* 10(9/10):354-356, 1974.

[1099] J. H. Wilkinson. Turing's work at the National Physical Laboratory and the construction of Pilot ACE, DEUCE, and ACE. In *A History of Computing* *in the Twentieth Century: A Collection of Essays,* N. Metropolis, J. Howlett, and Gian-Carlo Rota, editors, Academic Press, New York, 1980, pages 101–114.

[1100] J. H, Wilkinson. The state of the art in error analysis. In *NAG Newsletter 2/85,* Numerical Algorithms Group, Oxford, UK, November 1985, pages 5–28.

[1101] J. H. Wilkinson. Error analysis revisited. *IMA Bulletin,* 22(11/12):192-200, 1986.

[1102] J. H. Wilkinson and C. Reinsch, editors. *Linear Algebra,* volume II of *Handbook for Automatic Computation.* Springer-Verlag, Berlin, 1971. ix+439 pp. ISBN 3-540-054146.

[1103] James H. Wilkinson. The perfidious polynomial. In *Studies in Numerical Analysis, G.* H. Golub, editor, volume 24 of *Studies in Mathematics,* Mathematical Association of America, Washington, DC, 1984, pages 1–28.

[1104] M. R, Williams and Martin Campbell-Kelly, editors. *The Early British Computer Conferences,* volume 10 of *Charles Babbage Institute Reprint Series for the History of Computing.* MIT Press, Cambridge, MA, USA, 1989. xvi+508 pp. ISBN 0-262-23136-0.

[1105] S. Winograd. A new algorithm for inner product. *IEEE Trans. Comput.,* C-18:693-694, 1968.

[1106] S. Winograd. On multiplication of 2 × 2 matrices. *Linear Algebra Appl.,* 4: 381-388, 1971.

[1107] Jack M. Wolfe. Reducing truncation errors by programming. *Comm. ACM,* 7(6):355–356, 1964.

[1108] Philip Wolfe. Error in the solution of linear programming problems. In Error *In Digital Computation,* Louis B. Rall, editor, volume 2, Wiley, New York, 1965, pages 271-284.

[1109] Stephen Wolfram. *Mathematical: A System for Doing Mathematics by Computer.* Second edition, Addison-Wesley, Reading, MA, USA, 1991. xxii+961 pp. ISBN 0-201-51507-5.

[1110] Michael Woodger. The history and present use of digital computers at the National Physical Laboratory. *Process Control and Automation,* pages 437–442, November 1958. Reprinted in [186, pp. 125-140].

[1111] H. Woźniakowski. Numerical stability for solving nonlinear equations. *Numer. Math.,* 27:373–390, 1977.

[1112] H.Woźniakowski.. Numerical stability of the Chebyshev method for the solution of large linear systems. *Numer. Math.,* 28:191-209, 1977.

[1113] H.Woźniakowski.. Roundoff-error analysis of iterations for large linear systems. *Numer. Math.,* 30:301-314, 1978.

[1114] H. Woźniakowski. Roundoff-error analysis of a new class of conjugate-gradient algorithms. *Linear Algebra Appl.,* 29:507-529, 1980.

[1115] Margaret H. Wright. Interior methods for constraind optimization. In *Acta Numerics*, Cambridge University Press, Cambridge, UK, 1992, pages 341–407.

[1116] Stephen J. Wright. A collection of problems for which Gaussian elimination with partial pivoting is unstable. *SIAM J. Sci. Statist. Comput.*, 14(1):231–238, 1993.

[1117] P. Y. Yalamov. Roundoff errors and graphs. Manuscript, 1994.

[1118] E. L. Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman–Morrison–Woodbury formula. *SIAM J. Sci. Statist. Comput.*, 7(3):507-513, 1986.

[1119] J. M. Yohe. Implementing nonstandard arithmetics. *SIAM Rev.*, 21(1):34-56, 1979.

[1120] J. M. Yohe. Software for interval arithmetic: A reasonably portable package. *ACM Trans. Math. Software,* 5(1):50–63, 1979.

[1121] J. M. Yohe. Portable software for interval arithmetic. *Computing,* Suppl. 2: 211-229, 1980.

[1122] David M. Young. *Iterative Solution of Large Linear Systems.* Academic Press, New York, 1971. xxiv+570 pp. ISBN 0-12-773050-8.

[1123] David M. Young. A historical overview of iterative methods. *Comput. Phys. Comm.,* 53:1-17, 1989.

[1124] Gideon Yuval. A simple proof of Strassen's result. *Inform. Process. Lett.,* 7 (6):285-286, 1978.

[1125] Adam T. Zawilski. Numerical stability of the cyclic Richardson iteration. *Numer. Math.,* 60:251-290, 1991.

[1126] Hongyuan Zha. A componentwise perturbation analysis of the *QR* decomposition. *SIAM J. Matrix Anal. Appl.*, 4(4):1124–1131, 1993.

[1127] Hongyuan Zha. Problem 10312. *Amer. Math. Monthly*, 100(5):499, 1993.

[1128] G. Zielke. Report on test matrices for generalized inverses. *Computing,* 36: 105-162, 1986.

[1129] Gerhard Zielke. Some remarks on matrix norms, condition numbers, and error estimates for linear equations. *Linear Algebra Appl.,* 110:29–41, 1988.

[1130] K. Ziętak. On a particular case of the inconsistent linear matrix equation *AX + YB = C. Linear Algebra Appl.*, 66:249-258, 1985.

[1131] Abraham Ziv. Relative distance-An error measure in round-off error analysis. *Math. Comp.,* 39(160):563–569, 1982.

[1132] Abraham Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Software,* 17(3):410–423, 1991.

[1133] Abraham Ziv. Converting approximate error bounds into exact ones. *Math. Comp.,* 64(209):265-277, 1995.

[1134] Zahari Zlatev, Jerzy Wasniewski, and Kjeld Schaumburg. Condition number estimators in a sparse matrix software. *SIAM J. Sci. Statist. Comput.,* 7(4): 1175–1189, 1986.

Name Index

Science is organized knowledge.
— HERBERT SPENCER, *Essays On Education* (1861)

A suffix "t" after a page number denotes a table, "f" a figure, "n" a footnote, and "q" a quotation.

Aasen, Jan Ole, 226
Abdelmalek, Nabih N., 385, 410
Abramowitz, Milton, 35
Acton, Forman S., 35, 203 q, 285 q
Adams, Duane A., 113
Aggarwal, Vijay B., 84
Ahac, Alan A., 198
Ahlberg, J. H., 167
Aitken, A. C., 383
Albers, Donald J., 513 n
Alefeld, Göltz, 486
Alexopolous, Aristides G., 54
Allen, Jr., Richard C., 84
Almacany, Montaha, 425 q
Aluru, Srinivas, 459
Alvarado, Fernando L., 165
Amato, James J., 323
Ames, William F., 32
Amodio, Pierluigi, 197
Anda, Andrew A., 385
Anderson, E., 289 q
Anderson, T. W., 527
Ando, T., 196
Arioli, Mario, 143, 304, 328, 343, 410, 411, 421, 422
Arnold, William F., 323
Ashenhurst, R. L., 489
Asplund, Edgar, 305
Atanasoff, John V., 151 q
Axelsson, Owe, 343

Babuška, Ivo, 100
Bachelis, Boris, 61
Bai, Zhaojun, 322, 352, 527
Bailey, David H., 448, 460, 461, 471, 482, 491 q, 505
Baksalary, J. K., 323, 324
Bane, Susanne M., 461
Ballester, C., 441
Bank, Randolph E., 257
Bareiss, E. H., 53, 191
Bargmann, V., 186
Barlow, Jesse L., 33, 51, 53, 191, 193, 301, 385, 411, 412
Barnett, S., 323
Barone, John L., 391 q, 410
Barrett, Geoff, 59
Barrlund, Anders, 194, 198, 224
Bartels, R. H., 312
Bartels, Sven G., 141, 304, 441
Barwell, Victor, 226
Bauer, F. L., 57, 83, 119, 126, 139, 146, 148, 149, 191, 284, 551
Beam, Richard M., 527
Beaton, Albert E., 391 q, 410
Bell, E. T., 281 q
Bellman, Richard, 324
Benford, Frank, 51
Benôit, Commandant, 224

Benschop, N. F., 327
Berman, Abraham, 147, 579 q, 580
Bhatia, Rajendra, 322, 382
Bini, Dario, 441, 456, 457
Birkhoff, Garrett, 32, 281, 483
Bischof, Christian H., 299, 301, 370, 384
Björck, Åke, 83, 242, 361q, 379, 385, 386, 388, 397, 399, 400, 402, 403, 409-411, 422, 423, 434, 436, 441, 570
Bjørstad, Petter, 461
Blanch, G., 507
Bliss, B., 488
Blue, James L., 502
Bodewig, E., 188
Bohlender, Gerd, 98
Bohte, Z., 183
Boley, Daniel, 243
Bollen, Jo A. M., 328
Bondeli, S., 525
Boros, T., 441
Borwein, J. M., 491 q
Borwein, P. B., 491 q
Bowden, B. V., 203q
Bowdler, H. J., 196
Boyd, David W., 291, 293, 304
Boyle, Jeff, 51
Brent, Richard P., 51, 162, 164, 448, 451, 453, 460, 486, 504, 508
Briggs, William L., 470
Brightman, Tom, 60
Brown, W. S., 498, 501
Brunet, Marie-Christine, 53, 488
Buchan, John, 62 q
Buchanan, James L., 165
Buchholz, W., 60
Bukhberger, B., 305
Bulirsch, R., 83, 84, 190
Bunch, James R., 141, 146, 149, 219-221, 225, 226, 231 q
Buoni, John J., 198
Burgmeier, James W., 84
Businger, Peter A., 146, 193, 410
Butcher, J. C., 100
Byers, Ralph, 305, 320, 323, 324, 353, 567
Caffney, John, 520
Calvetti, D., 440442
Calvin (and Hobbes), 473 q
Calve, M. P., 32
Campbell, S. L., 337
Campbell-Kelly, Martin, 245 q
Canuto, Claudio, 32
Cao, Wei-Lu, 305
Caprani, Ole, 100
Cardano, Geronimo, 483
Carr III, John W., 57
Carter, Russell, 496
Cauchy, Augustin-Louis, 516
Cayley, Arthur, 446
Chaitin-Chatelin, Françoise, 53, 358, 488
Chan, Raymond H., 471
Chan, Tony F., 13, 33, 146, 147, 386
Chandrasekaran, Shivkumar, 136, 141, 386
Chang, Xiao-Wen, 225
Chartres, Bruce A., 190
Chatelin, Françoise, see Chaitin-Chatelin, Françoise
Choi, Man-Duen, 513 q, 526
Cholesky, André-Louis, 224
Christiansen, Søren, 146
Chu, Eleanor, 193
Chu, King-wah Eric, 323
Cipra, Barry A., 32
Clasen, B.-I., 284
Clenshaw, C. W., 32, 53, 113
Cline, Alan K., 289q, 297, 299, 411, 480
Cline, R. E., 422
Clinger, William D., 61
Cody, Jr., William J., 39q, 54, 55, 59, 60, 495, 497, 499, 504
Cohen, A. M., 180, 522
Concus, P., 257
Corm, Andrew R., 299
Conte, Samuel D., 190
Cooley, James W., 470
Coomes, Brian A., 32
Coonen, Jerome T., 59, 495
Cope, J. E., 145
Coppersmith, Don, 448

Corless, Robert M., 32
Cortés, Joaquín, 197
Cottle, Richard W., 224
Cox, M. G., 32
Crout, Prescott D., 195
Cryer, Colin W., 180, 181, 196
Curtis, A. R., 197
Cybenko, George, 225

Dahlquist, Germund, 83, 160
Daniel, J. W., 385
Datta, Karabi, 315
Davis, Philip J., 32, 100, 471
Dax, Achiya, 226, 337
Day, Jane M., 197
de Boor, Carl, 32, 190, 196
de Jong, Lieuwe Sytse, 33
de Rijk, P. P. M., 410, 411
Dekker, T. J., 57, 93, 284, 504
del Ferro, Scipione, 483
Demeure, Cédric J., 441
Demmel, James W., 47, 53, 60, 61, 84, 126, 140, 141, 143, 149, 165, 198, 207, 208, 224, 243, 248, 250-253, 257, 304,308, 322, 352, 417, 422, 486, 495, 499, 504, 527, 536
Dennis, Jr., J. E., 32, 328, 477, 478
Descloux, J., 343
Dhillon, Inderjit, 60, 536
Diamond, Harold G., 62
Dixon, John D., 300
Dongarra, Jack J., 188, 195,225,231 q, 257, 499, 581 q
Doolittle, Myrick Hascall, 195
Dorn, William S., 83, 100, 113
Douglas, Craig C., 460, 575
Douglas, Jr., Jim, 32, 188
Doyle, Sir Arthur Conan, 289 q
Drake, J. B., 261 q
Drmač, Zlatko, 225
Du Croz, Jeremy J., 265, 272, 273, 284
Dubrulle, Augustin A., 285, 511
Duff, Iain S., 143, 195, 200, 225, 226, 257, 304, 343, 410, 411, 527
Duncan, Martin, 96
Dunham, C. B., 425q
Dwyer, Paul S., 169 q

Eckart, Carl, 126
Edelman, Alan, 63, 162, 180, 181, 197, 198, 287, 386, 483n, 518, 518 q, 527, 534
Eirola, Time, 33
Eldén, Lars, 412
Eldersveld, Samuel K., 257
Elfving, Tommy, 441
Emel'yanenko, G. A., 305
Enright, Wayne H., 33
Erisman, A. M., 190, 193, 200
Espelid, Terje O., 100

Faddeeva, V. N., 195
Fairgrieve, Thomas F., 61, 530
Fan, Ky, 386
Farebrother, R. W., 409
Farkas, I., 113
Farnum, Charles, 497
Fateman, Richard J., 59
Feingold, David G., 257
Feldstein, A., 62
Ferguson, H. R. P., 461, 482
Ferguson, Jr., Warren E., 49, 60
Ferng, William R., 301
Fike, C. T., 114
Fischer, Patrick C., 460
Flannery, Brian P., 479, 490, 507
Fletcher, R., 32, 147, 149, 225, 226
Forsgren, Anders, 225
Forsythe, George E., 32, 33, 35, 52, 57, 84, 95, 138, 146, 164, 190, 196, 197, 235, 241, 242, 245 q, 261 q, 262, 282, 305, 325 q, 491 q, 526
Foster, Leslie V., 178, 411
Foulser, David E., 146, 147
Fox, L., xxviiq, xxviii, 35, 117 n, 188-190
Frayssé, Valérie, 53, 358
Friedland, Shmuel, 348, 359
Funderlic, R. E., 198

Gaches, J., 132, 145

Gahinet, Pascal M., 323
Gal, Shmuel, 61
Gallopoulos, E., 488
Gander, Walter, 525
Gantmacher, F. R., 172
Gardiner, Judith D., 323
Gardner, Martin, 127 q
Garner, Harvey L., 62
Gasca, M., 196
Gastinel, Noel, 123, 126
Gauss, Carl Friedrich, 1q, 195, 219, 325 q, 391
Gautschi, Walter, 425 q, 428, 429
Gautschi, Werner, 351
Gay, David M., 61
Geist, G. A., 261 q
Geman, Stuart, 518
Gentle, James E., 34
Gentleman, W. Morven, 113,384,385, 470, 576
George, Alan, 193, 224, 226
Geuder, James C., 190
Ghavimi, Ali R., 320, 322
Gill, Philip E., 32, 225, 226, 229, 422, 479
Gill, S., 92
Givens, Wallace J., 33, 67 q
Gluchowska, J., 411
Gohberg, I., 141, 440, 516
Goldberg, David, 39 q, 57, 93, 534
Goldberg, I. Bennett, 62
Goldstine, Herman H., 1 n, 33,52, 187, 196, 261q, 263, 517
Golub, Gene H., xxiv, 13, 27, 33, 146, 182, 190, 195, 223, 231 q, 257, 285, 301, 311, 312, 312 n, 327, 352, 384, 386, 388, 391, 392, 400, 409-412, 441, 580
Goodman, R., 62
Goodnight, James H., 285
Gould, Nicholas I. M., 181, 197
Govaerts, W., 242
Gragg, W. B., 297, 385
Graham, Ronald L., 87 q, 520 q
Grebogi, Celso, 32
Greenbaum, A., 328, 329
Gregory, Robert T., 514, 525
Grimes, Roger G., 305, 527
Grosse, Eric, 581 q
Gu, Ming, 352, 536
Gudmundsson, Thorkell, 301
Guggenheimer, Heinrich W., 287
Gulliksson, Mårten, 412
Gurwitz, Chaya, 32
Gustafson, John L., 459
Gustavson, F. G., 195

Hager, William W., 294, 304
Hall, Jr., Marshall, 179
Halmos, Paul R., 513 q
Hamada, Hozumi, 54
Hammarling, Sven J., 322, 326, 327, 385, 503
Hammel, Stephen M., 32
Hammer, Rolf, 487
Hamming, R. W., 57, 443
Handy, Susan L., 412
Hansen, Per Christian, 146, 386, 461
Hanson, Richard J., 385, 409, 410, 422, 503
Harris, P. M., 32
Harter, Richard, 459
Hartfiel, D. J., 146
Hearon, John Z., 315
Heath, Michael T., 261 q, 399
Hein, Piet, 36 q, 127q
Helvin, Marie, 166 q
Henderson, Harold V., 323, 490
Hennessy, John L., 39 n, 57
Henrici, Peter, 32, 52, 53, 67q, 84, 351
Henson, Van Emden, 470
Heroux, Michael, 460, 575
Herzberger, Jurgen, 486
Hewer, Gary, 323
Higham, Desmond J., 32, 126, 141, 149, 178, 179, 195, 345q, 348, 407, 441
Higham, Nicholas J., 72, 100, 141, 145, 164, 165, 178, 179, 195, 197, 223, 225, 242, 243, 248, 250-253, 257, 265, 272, 273, 284, 295, 300, 304-307, 322, 324, 328, 329, 354, 356, 358, 384,

386, 389, 407, 410, 411, 417, 422, 440, 443, 460, 461, 489, 508, 574, 577
Hilbert, David, 526
Hildebrand, F. B., 33, 39 q
Hocks, Matthias, 487
Hodel, A. Scottedward, 323
Hodges, Andrew, xxvii
Hoffman, A. J., 386
Hoffmann, Christoph M., 34
Hoffmann, W., 284, 385
Hooper, Judith A., 504
Horn, Roger A., 119, 150, 310, 348, 546, 551, 555, 558, 580
Horning, Jim, 491 q
Hotelling, Harold, 186, 187, 445 q
Hough, David, 47, 509
Householder, Alston S., 2, 117q, 126, 160, 172, 383, 410, 565
Hull, T. E., 52, 61, 488, 506, 530
Huskey, H. D., 188
Hussaini, M. Yousuff, 32
Hyman, M. A., 282

Ikebe, Yasuhiko, 303, 305, 307
Incertis, F., 512
Ipsen, Ilse C. F., 136, 141, 147, 385, 386
Iri, Masao, 53, 488
Isaacson, Eugene, 189, 257

Jalby, William, 385
Jankowski, M., 94, 241, 242
Jansen, Paul, 486
Jennings, A., 165
Jennings, L. S., 421
Johnson, Charles R., 119, 150, 287, 310, 348, 546, 551, 555, 558, 580
Johnson, Samuel, 675
Jones, Mark T., 226
Jones, William B., 507
Jordan, Camille, 284
Jordan, T. L., 409
Jordan, Wilhelm, 284

Kågstrom, Bo, 305, 320, 322-324
Kahan, William M. (Velvel), 1q, 29, 33, 34, 46, 47, 47q, 50, 50q, 59, 63-65, 75, 86, 92-95, 98, 113, 123, 126, 136, 161, 165, 169 q, 225, 243, 486, 490, 494, 496, 497, 499, 501, 502, 507
Kahaner, David K., 391 q
Kailath, T., 441
Kala, R., 323, 324
Kaniel, S., 226
Karasalo, Ilkka, 165
Karatsuba, A., 461
Karlin, Samuel, 523
Karlson, Rune, 404, 413
Karney, David L., 514, 525
Karp, A., 195
Karpinski, Richard, 56, 497
Kate, Tosio, 126
Kaufman, Linda, 221, 225, 226, 385
Keiper, Jerry B., 36
Keller, Herbert Bishop, 189,257,325 q
Kennedy, Jr., William J., 34
Kenney, Charles S., 301, 323, 525
Kerr, Thomas H., 225
Kiełbasiński, Andrzej, 84,94,224,242, 406, 411, 418, 422
Kincaid, D. R., 503
Kittaneh, Fuad, 525
Knight, Philip A., 328, 329, 354, 356, 358, 460, 461
Knuth, Donald E., xxiv, 54, 57, 58, 67q, 87q, 93, 94, 114, 461, 491 q, 520q, 526, 534
Koçak, Hüseyin, 32
Koltracht, I., 141, 517
Korner, T. W., 465q
Kornerup, Peter, 54
Kostlan, Eric, 518, 527
Kovarik, Z. V., 145
Kowalewski, G., 440
Krasny, Robert, 505
Kreczmar, Antoni, 460
Krogh, F. T., 503
Krol, Ed, 582
Kruckeberg, F., 198
Kubota, Koichi, 488

Kuczyński, J., 300
Kuki, H., 60
Kulisch, Ulrich W., 486, 487
Kuperman, I. B., 146

La Porte, M., 53
La Touche, Mrs., 87q, 88
Laderman, Julian, 450, 459
Lagrange, Joseph Louis, 219
Lancaster, Peter, 322
Lanczos, Cornelius, 490
Laratta, A., 421, 422
Larson, John L., 488
Larsson, S., 32
László, Lajos, 351
Laub, Alan J., 301, 320, 322, 323, 525
Läuchli, Peter, 388
Lawson, Charles L., 385, 409, 410, 422, 502
LeBlanc, E., 486
Lee, King, 460
Lehmer, D. H., 529
Lehoucq, R. B., 569
Lemeire, Frans, 165
Leuprecht, H., 98
LeVeque, Randall J., 13, 33
Lewis, John G., 33, 301, 305, 527
Li, T. Y., 285
Li, Xiaoye, 495
Linnainmaa, Seppo, 53, 83, 93, 100, 504
Linz, Peter, 100
Linzer, Elliot, 470
Liu, Joseph W. H., 224, 226
Longley, James W., 410
Lotstedt, Per, 412
Lotti, Grazia, 456, 457
Lu, Hao, 441
Lynch, Robert E., 32
Lyness, J. N., 32, 441
Lynn, M. Stuart, 327

Mac Lane, Saunders, 483
Macleod, Allan J., 197
Makhoul, John, 225
Malcolm, Michael A., 33, 95, 97, 196, 245q, 262, 305, 497, 509, 525
Marine, F., 461
Manteuffel, Thomas A., 165
Marovich, Scott B., 576
Marsaglia, George, 527
Martin, R, S., 196
Mascarenhas, Walter, 197
Mathias, Roy, 165, 223, 386, 515
Matsui, Shouichi, 53
Mattheij, R. M. M., 257
Matula, David W., 54, 57, 62
Mazzia, Francesca, 197
McCarthy, Charles, 146
McCracken, Daniel D., 83, 100, 113
McKeeman, William Marshall, 196, 241
McKenney, Alan, 322, 527
Meinguet, Jean, 224, 284
Mendelssohn, N. S., 286
Metcalf, Michael, 3
Metropolis, N., 489
Meurant, Gérard, 257, 305
Meyer, Jr., Carl D., 147, 337
Milenkovic, Victor J., 34
Miller, D. F., 322
Miller, Webb, 83, 115, 451, 454, 473q, 487, 488, 507, 517, 576
Miranker, Willard L., 486
Mirsky, L., 558
Moler, Cleve B., 33, 34, 59, 84, 95, 164, 190, 196, 197, 231 q, 235, 241, 243, 245q, 261q, 262, 282, 297, 304, 305, 323, 345q, 348, 359, 391q, 415q, 441, 503, 508, 511, 526, 579q
Møller, Ole, 92
Montgomery, D., 186
Moore, Ramon E., 486
Morrison, Donald, 503, 511, 579 q
Morton, K. W., 32
Mukherjea, Kalyan, 382
Müller, K. H., 113
Müller-Merbach, H., 32
Murakami, H., 483 n
Murray, Walter, 32,225,226,422,479

Nagy, James G., 471
Nash, Stephen G., 312, 324, 391q, 567
Nashed, M. Zuhair, 409
Neumaier, A., 61, 94
Neumann, M., 198
Newbery, A. C. R., 113
Newcomb, Simon, 51
Newman, Morris, 514, 520
Nickel, Karl, 94, 486
Nilson, E. N., 167
Nordio, Marcelo, 197
Notay, Yvan, 329

Oberaigner, W., 98
O'Cinneide, Colm Art, 147
Oettli, W., 135, 145
Ofman, Yu., 461
O'Leary, Dianne Prost, 305
Olesky, D. D., 198
Oliver, J., 113, 440
Olkin, Ingram, 527
Olshevsky, V., 440, 441
Olver, F. W. J., 53, 76, 84, 113, 190
Omladič, Matjaž, 225
Opfer, Gerhard, 441
Ortega, James M., 384
Osborne, M. R., 421
Ostrowski, A. M., 350, 546

Paige, C. C., 225, 379, 385, 388, 397, 411, 415q, 421, 422, 570
Palmer, John, 525
Palmer, Kenneth J., 32
Pan, Victor, 441, 448, 450, 459
Papadimitriou, Pythagoras, 386
Papadopoulos, Philip M., 323
Park, Haesun, 385
Parlett, Beresford N., xxiv, xxviii, 27, 34, 58, 87q, 187, 219, 225, 361 q, 383-385
Pasternak, Mary E., 488
Patashnik, Oren, 87 q, 520 q
Patrick, Merrell L., 226
Patterson, David A., 39 n, 57
Paxson, Vern, 61
Pelz, Richard, 505
Peña, J. M., 192, 196
Pereyra, Victor, 434, 436, 441
Peters, G., 113, 196, 279, 284, 287, 411
Peterson, Brian, 197
Philippe, Bernard, 385
Pichat, M., 98
Pierce, Daniel J., 301
Piessens, Robert, 32
Pinkus, Allan, 32, 190, 196
Plemmons, Robert J., 147, 198, 301, 337, 411, 422, 471, 579q, 580
Poljak, Svatopluk, 140
Polman, Ben, 257
Ponceleón, Dulce B., 226
Poromaa, Peter, 305, 320, 324
Pothen, Alex, 165
Powell, M. J. D., 225, 411, 474, 477
Power, Stephen, 224
Prager, W., 135, 145
Press, William H., 479, 490, 507
Priest, Douglas M., 33, 34, 58, 96-99, 102, 502, 504, 531
Pryce, J. D., 76, 242
Puglisi, Chiara, 371
Pukelsheim, Friedrich, 323
Puschmann, Heinrich, 197

Quarteroni, Alfio, 32
Quinlan, Gerald D., 96
Quinn, Thomas R., 96

Rabinowitz, Philip, 32, 100
Raimi, Ralph A., 52
Rail, Louis B., 409, 486, 488
Ramos, George U., 470
Rath, Wolfgang, 385
Ratz, Deitmar, 487
Ratz, H. C., 327
Razaz, M., 113
Reichel, Lothar, 113, 358, 440-442, 527
Reichelt, Mark W., 527
Reid, J. K., 3, 190, 193, 197, 200, 226, 411
Reinsch, C., xxviii, 284, 585 q
Reiser, John F., 58

Ren, Huan, 60
Rew, R. K., 289 q, 299, 480
Rice, John R., 33, 385
Richtmyer, Robert D., 32
Rigal, J. L., 132, 145
Robertazzi, T. G., 98
Roberts, J. D., 322
Rohn, Jiří, 128, 140, 141, 149
Romani, Francesco, 328
Rose, Donald J., 257
Rosenthal, Peter, 322
Ross, D. R., 97
Rowan, Thomas Harvey, 488, 489
Rubin, Donald B., 391 q, 410
Ruhe, Axel, 385
Ruiz, Daniel, 343
Rust, B. W., 145
Rutishauser, Heinz, 514, 520

Sameh, Ahmed H., 162, 164, 488
Samelson, Klaus, 57
Sande, G., 470
Sanz-Serna, J. M., 32, 33
Saunders, Michael A., 225, 226, 229, 257, 422
Sautter, Werner, 190
Scarborough, James B., 33, 58
Schaumburg, Kjeld, 305
Schelin, Charles W., 61
Scherer, R., 84
Schnabel, Robert B., 32
Schneider, Hans, 124, 359
Schonfelder, J. L., 113
Schreiber, Robert S., 165, 180, 245 q, 250–253, 257, 371, 384, 576
Schryer, N. L., 498
Schwartz, S. C., 98
Schwetlick, Hubert, 84,406,411,418, 422
Searle, Shayle R., 323, 490
Sha, Xuan-He, 450, 459
Shampine, Lawrence F., 32, 33, 84, 95, 527
Shannon, Claude E., 60
Shapiro, Alexander, 146
Shepherd, David, 59
Shinnerl, Joseph R., 225, 229
Shroff, Gautam M., 301
Shub, Michael, 147, 518
Simon, Horst D., 460
Skeel, Robert D., 36, 135, 146, 190, 192, 198, 235, 240, 241, 486
Slishman, Gordon, 460, 575
Smale, Steve, 2 n
Smith, David M., 505
Smith, Francis J., 113, 439
Smith, Jon M., 37
Smith, Robert L., 503, 509
Smith, Roger M., 460, 575
Smoktunowicz, Alicja, 94, 141, 149, 242, 411
Snyder, James N., 231 q, 241
Sokolnicka, Jolanta, 242
Sorensen, Danny C., 195, 225, 229, 257
Sørevik, T., 461
Spellucci, P., 32
Spencer, Herbert, 665
Spieß, J., 460
Spooner, David, 488
Steele, Jr., Guy L., 61
Stegun, Irene A., 35
Sterbenz, Pat H., 33, 34, 50, 57, 60, 489
Stewart, G. W. (Pete), xxiv, 75, 113, 119, 126, 131q, 139, 146, 147, 151 q, 164, 174, 195, 197, 198, 224, 231 q, 241, 242, 297, 305, 312, 324, 359, 381,382,385,386,392, 394, 407,409-411,504, 519,527, 580
Stewart, William J., 305
Steer, J., 83, 84, 119, 126, 127, 190
Stone, Betty Jane, 126
Storey, C., 323
Stoutemyer, David R., 489
Strakos, Zdenek, 329
Strang, Gilbert, 15, 124, 146, 471
Strassen, Volker, 446, 461, 462
Straus, E. G., 138, 146
Street, Anne Penfold, 179
Stummel, Friedrich, 34, 83, 191

Sun, Ji-guang, 119, 126, 131 q, 139, 146, 194, 198,209,224,382, 386,392,394,404,407, 409, 411, 413, 580
Sun, Xiaobai, 384
Swarm, W. H., 474
Swartzlander, Jr., Earl E., 54
Sweeney, D. W., 60
Swenson, J. R., 52
Sylvester, James Joseph, 309 q, 322, 446

Tang, Ping Tak Peter, 61, 499, 530
Tang, W. P., 441
Tao, Pham Dinh, 128, 304
Tartaglia, Niccolo, 483
Taussky, Olga, 513 q, 515
Teukolsky, Saul A., 479, 490, 507
Thacher, Jr., Henry C., 196
Thisted, Ronald A., 34
Thompson, Sir D'arcy Wentworth, 1q
Thron, W. J., 507
Tienari, Martti, 53
Todd, John, 126, 514, 520, 526, 527
Toh, Kim-Chuan, 483 n, 526
Torczon, Virginia J., 477-479
Tornheim, L., 180, 181
Traub, J. F., 440
Trefethen, Lloyd N., 6 q, 32, 169 q, 180, 328, 345 q, 346, 348, 352, 353, 355, 358, 483 n, 526, 527
Tremaine, Scott, 96
Tropp, Henry S., 491n
Trummer, Manfred R., 32, 346, 355, 358
Tsao, Nai-kuan, 84, 384
Tukey, John W., 60, 470
Turing, Alan Mathison, xxvii, xxviii, 33, 126, 131 q, 188, 284,485
  contributions in 1948 paper "Rounding-off errors...", 188, 284
Turnbull, H. W., 383, 523
Turner, Kathryn, 225
Turner, Peter R., 51, 53, 165
Tyrtyshnikov, Evgenij E., 526

Underhill, L. G., 527
Ungar, Peter, 461
Urabe, Minoru, 62
Uspensky, J. V., 490

Vajteršic, M., 461
van der Sluis, A., 137, 138, 198, 207, 391 q, 409
Van der Vorst, Henk A., 195, 225, 257, 329
Van Loan, Charles F., xxiv, 27, 141, 149, 182, 190, 195, 223, 225, 229, 231 q, 257, 299, 311, 312, 312n, 345q, 348, 352, 359, 370, 371, 384, 386, 388, 392, 409, 412, 465q, 466, 470, 580
van Veldhuizen, M., 196
van Wijngaarden, A., 501
Vanderbei, Robert J., 229
Varah, James M., 146, 167, 257, 258
Varga, Richard S., 160, 167, 257, 508
Vemulapati, Udaya B., 301, 412
Veselić, Krešimir, 225
Vetterling, William T., 479, 490, 507
Vieta, Franciscus, 483
Vignes, J., 53
Vitasek, Emil, 100
Viten'ko, I. V., 95
von Neumann, John, 33, 52, 186, 187, 196, 263, 517

Waite, William, 497, 499
Waldén, Bertil, 404, 413
Walker, Homer F., 32, 328
Wallis, Jennifer Seberry, 179
Wallis, W. D., 179
Wang, Johnson J. H., 198
Ward, Robert C., 285
Warming, Robert F., 527
Wasilkowski, G. W., 147
Wasniewski, Jerzy, 305
Wasow, Wolfgang R., 32
Watson, G. A., 146
Watterson, Bill, 473
Wedderburn, J. H. M., 117q

Wedin, Per-Åke, 392, 407, 409, 412, 422
Wegert, Elias, 353
Wei, Musheng, 388, 410
Weidner, Peter, 486
Weiss, N., 147
Wendroff, Burton, 189, 197
Werner, Wilhelm, 113
Westin, Lars, 305, 323
Westlake, Joan R., 514
Wette, Matthew R., 323
White, Jon L., 61
Wichmann, B. A., 499, 502
Wilkinson, J. H., xxi, xxiv, xxvii, xxviii, 14, 25, 31, 33, 34, 39q, 52, 56, 58, 67q, 73, 75, 84, 100, 103q, 104, 113, 117q, 145, 151 q, 164, 175, 177, 178, 180-182, 187-191, 195-197, 203 q, 207, 208, 224, 232, 241, 252, 279, 283-285, 287, 297, 326, 327, 343 q, 348, 361 q, 364, 365, 384, 385, 387, 391, 400, 409-411, 474, 485, 542, 585 q
  first program for Gaussian elimination, 195–196
  on the purpose of a priori error analysis, 203 q
  solving linear systems on desk calculator in 1940s, 187–188
  user participation in a computation, 31
Williams, Jack, 425 q
Wilson, Greg, 60
Winograd, Shmuel, 446, 448, 461
Wisniewski, John A., 488
Witzgall, C., 119, 126, 127
Wolfe, Jack M., 97
Wolfe, Philip, 32
Woodger, Michael, 169 q, 445 q
Woźniakowski, H., 32, 94, 147, 241, 242, 300, 328
Wrathall, Celia, 488
Wright, Margaret H., 32, 225, 479
Wright, Stephen J., 178

Yalamov, P. Y., 83
Yip, E. L., 490
Yohe, J. Michael, 486, 504
Yorke, James A., 32
Young, David M., 326 t, 350
Young, Gale, 126
Yuvzd, Gideon, 459

Zang, Thomas A., 32
Zawilski, Adam T., 328
Zehfuss, Johann Georg, 323
Zeller, K., 84
Zeng, Z., 285
Zha, Hongyuan, 167, 382
Zielke, Gerhard, 126, 527
Ziętak, K., 323
Ziv, Abraham, 60, 61, 83, 84
Zlatev, Zahari, 305

Subject Index

> *Knowledge is of two kinds.*
> *We know a subject ourselves,*
> *or we know where we can find information upon it.*
> — SAMUEL JOHNSON, Boswell's *Life of Johnson* (1775)

A suffix "t" after a page number denotes a table, "f" a figure, "n" a footnote, and "q" a quotation. Mathematical symbols and Greek letters are indexed as if they were spelled out. The solution to a problem is not indexed if the problem itself is indexed under the same term.

Aasen's method, 226, 229
absolute error, 4
absolute norm, 119
accuracy versus precision, 7, 33
ACRITH, 486
Aitken extrapolation, 101
alternating directions method, 477
approximation theory, references for rounding error analysis, 32
Augment precompiled, 486, 504, 506
augmented system matrix, 393
  scaling and conditioning, 402
Automatic Computing Engine (ACE), 56, 189t, 195, 343
automatic differentiation, 488
automatic error analysis, 473–490, *see* also interval analysis; running error analysis
  condition estimation, 480-481
  solving a cubic, 483–484
  Strassen's inversion method, 481-482
  using direct search optimization, 474-477

backward error, 7-8
  componentwise, 134, 141
    evaluating, 143
  componentwise relative, 134
  definition, 7
  least squares problem, 404-407, 413
  linear system
    Oettli-Prager theorem, 135
    Rigal-Gaches theorem, 14, 132
  Lyapunov equation, 316–317
  mixed forward-backward error, 8
  normwise, 132
  normwise relative, 132
  preserving symmetric structure, 149, 406–407
  structured, 141
  Sylvester equation, 313-316
  underdetermined system, 419, 423
backward error analysis
  development by Wilkinson, 33–34, 189
  in differential equations, 33
  motivation, 7
  not a panacea, 1q
  purpose, 71–72, 203 q

- backward stability
  - componentwise, 142
  - definition, 8
  - normwise, 142
- banded matrix, growth factor, 183
- Bartels-Stewart method, 311-313
- Bauer’s scaling theorem, 139, 146
- bilinear noncommutative matrix multiplication algorithm, 449–450
  - error analysis, 456–457
- binary-decimal conversion, 61-62
- bit, 60
- BLAS (Basic Linear Algebra Subprograms), 586-587
  - fast level 3, 460
  - level 2 extended precision extension, 506
  - xNRM2 (2-norm), 502–503, 510–511
- block algorithm
  - advantages of, 245 q
  - definition, 246
- block diagonal dominance, 251-255, 257
  - and block LU factorization, 252–255
  - definition, 251
- block LU factorization, 246-259
  - computation, 247
  - definition, 246
  - error analysis, 250-257
  - existence and uniqueness, 247
  - stability
    - for (point) diagonally dominant matrix, 255
    - for block diagonally dominant matrix, 251–255
    - for block tridiagonal matrix, 258
    - for symmetric positive definite matrix, 255–257
- Bunch-Kaufman factorization, 221-223
- Bunch-Parlett factorization, 219-220
- byte, 60

- calculator, displaying words on, 37
- cancellation, 10–11, 30
  - in summation, 91, 543
  - not a bad thing, 11
  - of rounding errors, 21–26
- Cauchy matrix, 516-517
  - inverse, 516
  - LDU factors, 516-517
- Cauchy-Schwarz inequality, 119
- CELEFUNT, 499
- chaos, references for rounding error, analysis, 32
- Chebyshev spectral differentiation matrix, 346, 355
- Cholesky factorization, 204
  - computation of, 205
  - conditions for success in floating point, 208-209
  - error analysis, 205–209
  - existence and uniqueness, 204
  - perturbation bounds, 209–210
  - semidefinite matrix
    - complete pivoting, 211
    - computation of, 210–211
    - error analysis, 214–218
    - existence and uniqueness, 210
    - perturbation theory, 211–214
    - termination criteria, 217–218
- chopping, 57
- circulant matrix, 468
- circulant system, error analysis for solution by FFT, 468-470
- CLAPACK, 588
- colon notation, 2–3
- companion matrix, 525-526
  - singular values, 525–526
- comparison matrix, 157
- compensated summation, 92–97
- complete pivoting, 170
  - early use of, 196
  - fallacious criticism of, 200
  - growth factor, 180-181, 197
    - conjecture proved false, 181
- complex arithmetic, error analysis, 78–80, 84
- complex number
  - division without overflow, 503-504, 509

    square root of, 36
componentwise relative error, 5
condition number
    distance to singularity and, 123, 126, 140
    estimation, 289–308
        counterexamples, 289 q, 294–296, 299, 304, 305
        counterexamples by direct search, 480-481
        for tridiagonal matrices, 301–303
        incremental, 301
        LAPACK estimator, 294-297, 480481
        LINPACK estimator, 297-299
        probabilistic methods, 300-301
    general theory, 33
    Hadamard, 281, 287
    minimizing by scaling, 136–139, 146
    of function, 9
    of linear system
        componentwise, 135
        normwise, 133
    of rectangular matrix, 392
    of square matrix, 121–123, 126
    of summation, 100
    Skeel's, 135
conjugate gradient method, 328, 329, 341
    circulant preconditioned, 471
continued fraction
    algorithms and error analysis, 507
    evaluating in IEEE arithmetic, 492-493
    running error bound, 85
convergent matrix, 348
conversion, binary-decimal, 61–62
correct significant digits, 4–5, 32
Cramer's rule, (instability of, 14-15, 34, 37
Cray computers
    adoption of IEEE arithmetic, 49
    arithmetic on, 39 q, 495
    puzzling results from Cray Y-MP and Cray 2, 496497
    UNICOS library, 448, 450
Crout's method, 174
CS decomposition, 388, 408
cubic equation
    Newton's method, 489-490
    stability of explicit formulae for roots, 483484
cyclic reduction, 197

denormalized numbers, see subnormal numbers
departure from normality (Henrici's), 351-352
determinant, 281-283
    computation of, 282–283
    condition number of, 287
    of upper Hessenberg matrix, 27–28
diagonal dominance, 181
    and block LU factorization, 255
    block, 251-255, 257
    bound for LU factors of tridiagonal matrix, 185
    growth factor, 181
    matrix inverse bound, 167
        for tridiagonal matrix, 303
diagonal pivoting method, 218-223
    complete pivoting and its stability, 219–220
    growth factor
        complete pivoting, 220
        partial pivoting, 222
    partial pivoting and its stability, 221-223
diagonally dominant matrix
    bound for inverse, 167
    growth factor for, 181
differential equations, see ordinary differential equations; partial differential equations, references for rounding error analysis
direct search optimization methods, 477-479
discretization error, 6
distance to singularity
    componentwise, 140

  - normwise, 123
- divided differences, 109-112
  - confluent, 430
- Doolittle's method, 173-174, 195
- double rounding, 48, 63, 541
- Drazin inverse, 336-337
- drift, in floating point arithmetic, 58
- dual norm, 119
- dual vector, 119
- dynamical systems, references for rounding error analysis, 32

- effective conditioning, 146
- EISPACK, 587
- ELEFUNT package, 499
- equilibration, 136, 138, 192
- error
  - absolute, 4
  - backward, see backward error
  - forward, see forward error
  - mixed forward-backward error, 8
  - relative, 4, 5
  - sources of, 5-6
- error analysis, see rounding error analysis
- ESSL library (IBM), 448, 450
- expm 1 function (e^x – 1), 34

- fan-in algorithm, 165
  - for summation, 88
  - for triangular system solution, 162–164
- fast Fourier transform, 465-471
  - Cooley-Tukey factorization of DFT matrix, 466
  - error bound, 467
  - for solving circulant systems, 468-470
- fast matrix multiplication, 445463
  - bilinear noncommutative algorithm, 449450
  - deriving methods, 459460
  - error analysis, 450-459
  - in the level 3 BLAS, 460
  - Miller's error results, 451
  - record exponent, 448
  - Strassen's method, 446448
    - Winograd's variant, 448, 455–456
  - 3M method for complex multiplication, 450
  - Winograd's method, 446
- fixed point arithmetic, 56
- fl operator (rounding), 42
- floating point arithmetic, 39-65
  - alternatives to, 53-54
  - banned from safety-critical systems, 499
  - binary-decimal conversion, 61-62
  - choice of base, 51, 60–61
  - compiler optimization, dangers of, 497
  - determining properties of, 497-498
  - drift in, 58
  - earliest subroutines, 39 q
  - formal $\boxed{\mathrm{op}}$ algebra, 58
  - fused multiply-add operation, 60, 65
  - IEEE arithmetic, see IEEE arithmetic
  - Language Independent Arithmetic (LIA-1), 502
  - model, 58, 501–502
    - Brown's, 498, 501-502
    - standard, 44
    - with underflow, 61
    - without guard digit, 49
  - multiple precision, 504–506
  - parameters for selected machines, 41t
  - parameters in software, specifying, 499–500
  - representation error, 51
  - rounding, see rounding
  - software issues, 491–512
  - speed of operations (relative), 60
  - subnormal numbers, 41, 47, 495
  - subtraction done exactly, 49–50
  - testing accuracy of, 54–56
  - testing correctness of, 498499
  - unit roundoff, 3, 42
  - wobbling precision, 43, 51
- floating point coprocessor, 47

floating point numbers
  characterization, 40
  normalized, 40
  spacing between, 41
  subnormal, 41, 47
  testing for equality, 495
flop, 3
Fortran 90, 3
  environmental inquiry functions, 498
  matmul, 460
forward error, 7–8
  definition, 7
  for linear system, 13
  mixed forward-backward error, 8
forward stability
  componentwise, 142
  definition, 10
  normwise, 142
Fourier matrix, 179
FPV (floating point verification) package, 498499
Frobenius norm, 120
fused multiply-add operation, 60, 65

γ_n (error constant)
  definition, 69
  properties, 74
Gauss-Jordan elimination, 275-281,284-285
  algorithm, 276
  error analysis, 277–281
Gauss-Seidel method, 325 q, 334
Gaussian elimination, 170–174, *see also* LU factorization
  a posteriori stability tests, 192–194
  complete pivoting, 170
  computer programs
    first, 195-196
    history of, 196
  connection with LU factorization, 171
  error analysis, 174–177
    history of, 186-191
  growth factor, 177–183, see also growth factor
  in ancient China, 195
  loop orderings, 195
  need for pivoting, 170
  on Hessenberg matrix, 27–28
  partial pivoting, 170, 173
  pessimism of its accuracy in 1940s, 186–187
  row and column scaling, 191–192
  threshold pivoting, 200
  use by Gauss, 195
  versus Cramer's rule, 14–15
  without pivoting, instability of, 17
Gelfand's problem, 51
geometric computation, accuracy of algorithms in, 34
Givens rotation, 371
  disjoint rotations, 373-375, 387
  fast, 385
gradual underflow, 47, 61
Gram-Schmidt method, 376-381
  classical
    algorithm, 377
    error analysis, 378–379, 381
  modified
    algorithm, 377
    connection with Householder QR factorization, 361q, 379, 385
    error analysis, 378–381
    error analysis for application to LS problem, 396-397
    stability, 27
  reorthogonalization, 385
growth factor, 177-183
  a posteriori estimates for, 193
  define using exact or computed quantities?, 177, 196
  for banded matrix, 183
  for complete pivoting, 180-181
  for diagonal pivoting method, 220, 222
  for diagonally dominant matrix, 181
  for partial pivoting, 177-183
    large growth in practical problems, 178

  for random matrices, 196-197
  for tridiagonal matrix, 183
  for upper Hessenberg matrix, 182
  lower bound for, 179
  maximization by direct search, 475–476
  numerical maximization for complete pivoting, 181, 197
  statistical model of, 180
guard digit, 48
  test for, 56

Haar distribution, random orthogonal matrix from, 519–520
Hadamard condition number, 281,287
Hadamard matrix, 128, 179, 181, 201
Hadamard's inequality, 287
Harwell-Boeing sparse matrix collection, 527
Heron's formula, 50
Hessenberg matrix
  determinant of, 27-28, 34, 282-283
  Gaussian elimination, 27-28, 34
  growth factor for, 182
Hewlett-Packard HP 48G calculator
  condition estimator, 304
  exhausting its range and precision, 17–18
hidden bit, 45
Hilbert matrix, 514-517
  Cholesky factor, 515
  inverse, 515
Holder inequality, 118, 119
Homer's method, 104-115
  for derivatives, 106-109
  for rational function, 29
  running error bound, 105–106, 113
Hough's underflow story, 509-510
Householder transformation, 362-363
  aggregated (WY representation), 370-371
  block, 384
  error analysis, 364–369
  history of, 383
  in QR factorization, 363–364
Hyman's method, 34, 282-283, 287

IBM, ESSL library, 448, 450
IEEE arithmetic, 43, 4548
  double rounding, 48, 63, 541
  exception handling, 46, 493–495
  exceptions, 4647
  exploiting in software, 492–495
  extended formats, 47
  gradual underflow, 47
  implementation using formal methods, 59-60
  ∞,46–47, 492, 495
  NaN, 46, 492, 495
  parameters, 41 t, 45
  recommended auxiliary functions, 495
  rounding modes, 46
  signed zeros, 46
  Standard 754, 45
  Standard 854, 48
  subnormal numbers, 47, 495
index of a matrix, 336
∞ (IEEE arithmetic), 46–47, 492, 495
inner product
  error analysis, 68–71
  in extended precision, 70
  reducing constant in error bound, 69
Intel Pentium chip, division bug, 59
Internet, 582
interval analysis, 46, 198, 485–487
  dependencies, 485
  fallibility, 487
  Gaussian elimination, 485-486
  super-accurate inner product, 486
interval arithmetic, *see* interval analysis
inverse iteration, 27
inverse matrix, 26 1–287
  bound using diagonal dominance, 167
  error analysis
    for Gauss-Jordan elimination, 277–281
    for LU factorization, 270-275
    for triangular matrix, 265-270
  high accuracy computation of, 284
  in solving $Ax = b$, stability, 262

  left and right residuals, 263-264
  perturbation theory, 140
  times for computation on early computers, 276 t
  triangular
    bounds for, 159-161
    error analysis, 265–270
  why not to compute, 262
involutary matrix, 521
iterative methods, see also stationary iterative methods
  dates of publication, 326 t
  error analysis, 329-341
    survey of, 327–329
  stopping criteria, 341–342
iterative refinement, 30, 188, 231–244, 497
  backward error analysis, 235-240
  behaviour
    with GEPP, 239
    with QR factorization, 375–376
  condition number estimate from, 243
  for least squares problem, 399-403, 410
  for Vandermonde system, 438–440
  forward error analysis, 232-235
  in fixed precision, 234
  in mixed precision, 234
  LAPACK convergence test, 242
  practical issues, 242–243

Jacobi method, forward error analysis, 332–334
Jordan canonical form, 346-348

Kahan matrix, 161, 214
  second smallest singular value, 167
Kreiss matrix theorem, 353
Kronecker product, 310, 323

LANCELOT, 181, 197
LAPACK, 587-589
  block and partitioned LU factorization, 258
  Cholesky factorization, 226-227
  condition number estimation, 294–297, 306
  diagonal pivoting method, 227
  forward error bound for linear systems, 144
  iterative refinement, 242–243
  least squares problem, 412
  LU factorization, 192, 198-199, 258
  matrix l-norm estimator, 294–297, 480–481
  matrix inversion, 285
  QR factorization, 386-387
  Sylvester equation, 324
  test matrix generation, 527
  triangular systems, 166
  2 x 2 problems, solving, 500-501
  underdetermined system, 423
  xLAMCH for determining machine parameters, 498
Latin, neoclassic, publishing papers in, 470
LDL^T factorization, 205
  block, 218
least significant digit, 40
least squares problem, 391-413
  augmented system matrix, 393
    scaling and conditioning of, 402
  backward error, 404-407, 413
  constrained, 412
  iterative refinement, 399–403, 410
  Longley test problem, 410
  modified Gram–Schmidt, error analysis, 396–397
  normal equations
    error analysis, 397–399
    versus QR factorization, 399
  perturbation theory, 392–394
  QR factorization, error analysis, 395-396
  seminormal equations, 403–404
  weighted, 411–412
Leja ordering, 111, 113, 115, 438
level index arithmetic, 53
linear system
  large dense, in applications, 198
  perturbation theory, 131–150

  - practical forward error bounds, 143–144
  - records for largest solved, 199 t
  - scaling before Gaussian elimination, 191–192, 197–198
  - times for solution on early computers, 189 t

LINPACK, 587
  - Cholesky factorization of semidefinite matrix, 217
  - condition estimator, 297–299, 305
  - iterative refinement, 242
  - LU factorization, 192
  - matrix inversion, 267, 271–274
  - tridiagonal system solution, 306

logarithmic distribution of numbers, 51, 53

Longley test problem, 410

LU factorization, 169-201, *see also* Gaussian elimination
  - a posteriori stability tests, 192–194
  - block, see block LU factorization
  - complete pivoting, 170
  - Crout's method, 174
  - determinantal formulae for factors, 172
  - Doolittle's method, 173-174
  - error analysis, 174–177
    - history of, 186-191
  - existence and uniqueness, 171-172
  - for nonsymmetric positive definite matrix, 223–224
  - growth factor, 177-183, see also growth factor
  - loop orderings, 195
  - of Hessenberg matrix, 27–28
  - of tridiagonal matrix, 184
  - partial pivoting, 170, 173
  - partitioned, error analysis of, 248–250
  - perturbation bounds, 194
  - row and column scaling, 191–192
  - stability for *M*-matrix, 198
  - versus Cramer's rule, 14–15
  - without pivoting, instability of, 17

Lyapunov equation
  - backward error, 316-317
  - discrete-time, 322

M-matrix, 580
  - stability of LU factorization, 198
  - triangular, 157, 159, 160

machar code, 497498

machine epsilon, 41

magic square matrix, p norm of, 127

mantissa, 40

Maple, 6, 181, 506

Markov chain, perturbation analysis for, 146

Mathematical, 6, 181, 506

MATLAB, 3, 43, 583
  - condest, 304, 480
  - fft, 468
  - inv, 271
  - rand, 518
  - randn, 518
  - rcond, 305, 480, 489
  - roots, 483
  - special matrices, 514
  - Symbolic Math Toolbox, 3, 521
  - Test Matrix Toolbox, 591–594

matrix
  - block diagonally dominant, 251
  - Cauchy, 516-517
  - circulant, 468
  - companion, 525–526
  - comparison, 157
  - condition number, 121–123, 126, 392
  - confluent Vandermonde-like, 429
  - convergent, 348
  - diagonally dominant, 181
  - distance to singularity, 123, 140
  - Drazin inverse, 336-337
  - Fourier, 179
  - Hadamard, 128, 179, 181, 201
  - Hilbert, 514–517
  - inversion, 261–287, see also inverse matrix
  - involuntary, 521

  Kahan, 161, 167, 214
  M-matrix, 580
  magic square, 127
  moment, 520
  nonsymmetric positive definite, 223
  Pascal, 520-524
  powers of, 345–359, *see also* powers of a matrix
  pseudo-inverse, 392, 409, 412, 416
  random, 517–520
  randsvd, 519–520
  second difference, 525
  semiconvergent, 337
  Sylvester's introduction of term, 309
  symmetric indefinite, 218
  symmetric positive definite, 204
  symmetric positive semidefinite, 210
  symmetric quasidefinite, 229
  test, 513–528
  totally nonnegative, 176
  tridiagonal, 183–186
    Toeplitz, 524–525
  Vandermonde, 426
  Vandermonde-like, 429
  vet-permutation, 319, 323
matrix multiplication
  backward error, 85
  error analysis, 76–78, 85
  fast methods, 445-463
matrix norm, see norm
meaningless answers, why you might get them, 35
misconceptions, of floating point arithmetic, 31
mixed forward–backward error, 8
modified Gram–Schmidt method, see Gram–Schmidt method, modified
moment matrix, 520
monotone norm, 119
Moore-Penrose conditions, 412
most significant digit, 40
multi-directional search method, 477–479
multiple precision arithmetic, 504–506
  Bailey's package MPFUN, 505
  Brent's package, 486, 504
mutation testing, 517

NAG Library, 583-584
  LAPACK in, 588
  machine constants, 500
NaN (not a number), 46, 492, 495
Nelder-Mead simplex method, 479
netlib, 582–583
nonlinear equations, references for rounding error analysis, 32
nonsymmetric positive definite matrix, 223
  LU factorization, stability of, 223-224
norm, 117–129
  $\|\cdot\|_{\alpha,\beta}$, explicit formulae for, 128
  absolute, 119
  consistent, 120
  dual, 119
  Frobenius, 120
  Holder inequality, 118
  matrix, 120–124
    norm equivalence constants, 122t
  matrix p-norm, 124–126
    of magic square matrix, 127
  monotone, 119
  subordinate matrix, 120, 121
  2-norm, evaluation without overflow, 502-503
  unitarily invariant, 121
  vector, 118–119
    norm equivalence constants, 121t
normal equations, error analysis, 397–399
notation, explanation of, 2–3, 73–76
0°, definition, 64
NPSOL, 197
numerical analysis, definition, 6
numerical radius, 350
numerical stability
  definition, 8, 33
  for linear equation solvers, 141-142

Oettli–Prager backward error theorem, 135
optimization, references for rounding error analysis, 32
ordinary differential equations
  accuracy of mesh point formation, 101
  backward error in, 33
  Euler's method with compensated summation, 95
  references for rounding error analysis, 32
outer product, error analysis, 71
overflow, 18, 42
  avoiding, 502–504

p-norm power method, 291–294, 304
parallel prefix operation, 165
paranoia code, 497498
partial differential equations, references for rounding error analysis, 32
partial pivoting, 170, 173
  early use of, 196
  growth factor, 177-183
    large growth in practical problems, 178
  threshold pivoting, 200
partitioned algorithm, definition, 246
partitioned LU factorization, error analysis, 248–250
Pascal matrix, 520-524
  Cholesky factor, 521
  inverse, 522
  total positivity, 523
Patriot missile software problem, 506–507
Pentium chip, division bug, 59
performance profile, for LAPACK norm estimator, 296–297
perturbation theory
  by calculus, 144-145
  linear systems, 131-150
  statistical, 147, 149-150
  Sylvester equation, 318-320
pi (π), high precision calculation as computer test, 491q
polar decomposition, 386, 389
polynomials, 103–1 15, see also Horner's method
  divided differences, 109–1 12
  fast evaluation schemes, 114, 115
  Newton form, 109-112
PORT library, machine constants, 500
portability of software, 499-502
positive definite matrix, see nonsymmetric positive definite matrix; symmetric positive definite matrix
power method, 26
  for matrix l-norm estimation, 294-297
  for matrix p-norm estimation, 291-294, 304
powers of a matrix, 345–359
  behaviour of stationary iteration, 358
  departure from normality, 351-352
  hump, 348-349
  in exact arithmetic, 346–353
  in finite precision arithmetic, 353–358
  pseudospectrum, 352, 356-358
  role of spectral radius, 346–348
precision
  effect of increasing, 19-21
  versus accuracy, 7, 33
program verification, applied to error analysis, 488
pseudo-inverse, 392, 409, 412, 416
ε-pseudospectral radius, 352
pseudospectrum, 352, 356-358
  of companion matrix, 526
Pythagorean sum, 503, 511-512

QR factorization, 361-389
  column pivoting, 387
  Givens, 371-373
    cancellation of errors in, 24–26
    error analysis, 373–375
  Householder, 363-364
    error analysis, 368–369

    error analysis for application to LS problem, 395-396
    error analysis for partitioned (WY representation), 370-371
  iterative refinement for linear system, 375–376
  perturbation theory, 381–383
  rank-revealing, 386
quadratic equation, solving, 1 1–12, 33
quadrature
  accuracy of grid formation, 101
  error bound for evaluation of rule, 86
  references for rounding error analysis, 32

random matrices, 5 17–520
  condition number of, 518
  expected number of real eigenvalues, 518
  orthogonal, 519, 527
  spectral radius of, 518
  tend to be well conditioned, 518
  2-norm of, 518
  with given singular values, 519–520
randsvd matrix, 519–520, 527
range reduction, 54
RCOND condition estimator (LINPACK, MATLAB), 305, 480–481, 489
relative error, 4, 5
  componentwise, 5
relative error counter, <k>, 75
relative precision, 76
relative residual, 14
research problems, 102, 115, 201, 229, 244, 287, 308,324,359,423, 442, 463, 490, 527, 528
residual, relative, 14
Riccati equation, algebraic, 322
Rigal–Gaches backward error theorem, 132
rounding, 4, 42
  dealing with ties, 42, 58
  to even versus to odd, 58
rounding error analysis
  automatic, 473–490
  demystified, 82–83
  graphs in, 83
  model
    standard, 44
    with underflow, 61
    without guard digit, 49
  notation, 73–76
  ordering of operations, effect of, 77, 154
  purpose of, 71–72, 203 q
  statistical approach, 52–53
rounding errors
  accumulation of, 16
  are not random, 29, 52
  beneficial effects of, 26-27
  cancellation of, 21–26
  in subtraction, 49–50
  statistical assumptions on, 52–53
rules of thumb
  condition for computed powers of matrix to converge to zero, 358
  forward error related to backward error and condition number, 10
  relative speed of floating point operations, 60
  square root of constants in error bound, 52
Runge-Kutta method, 92, 100
running error analysis, 73, 489
  for continued fraction, 85
  for Homer’s method, 105–106, 113
  for inner product, 72–73

sample variance, see variance
ScaLAPACK, 588
scaling a linear system before Gaussian elimination, 191-192, 197–198
scaling to minimize the condition number, 136-139, 191
Schur complement, 219, 224, 247, 252
  perturbation bounds for symmetric positive semidefinite matrix, 212–218

second difference matrix, 525
semiconvergent matrix, 337
seminormal equations
  for least squares problem, 403-404
  for underdetermined system, 417
separation (sep), of two matrices, 318
Sherman-Morrison formula, 197, 490
significance arithmetic, 489
significant digits
  correct, 4–5, 32
  least and most significant, 40
singular value decomposition (SVD), 580
software
  avoiding underflow and overflow, 502-504
  effects of underflow, 504
  issues in floating point arithmetic, 491-512
  portability, 499-502
  specifying arithmetic parameters, 499-500
  specifying numerical constants, 501
SOR method, forward error analysis, 334
square root, of complex number, 36
stable algorithms, designing, 30-31
stationary iterative methods, 325–343
  and powers of a matrix, 358
  backward error analysis, 334-336
  forward error analysis, 329-334
    singular systems, 338–341
  Jacobi method, 332-334
  scale independence, 331
  singular systems, theory for, 336–338
  SOR, 334
  stopping criteria, 341–342
statistics, *see also* variance
  computational references, 34
sticky bit, 46
Strassen's method, 446-448
  accuracy compared with conventional multiplication, 454-455
  error analysis, 452–456
  error versus operation count, 16
  for inversion, 461-462, 481-482
  implementation issues, 460
  Winograd's variant, 448,455-456
subdifferential, of a vector norm, 291
subgradient, 292
subnormal numbers, 41, 47, 495
summation, 87–102
  choice of method: summary, 98–100
  compensated and applications, 92–97
  condition number, 100
  criterion for minimizing error, 90
  distillation algorithms, 98
  doubly compensated, 96-97
  error analysis, 89–92
  insertion method, 88
  pairwise (fan-in), 88
  recursive, 88
    ordering in, 19, 90-91
  statistical error estimates, 98
SVD (singular value decomposition), 580
Sylvester equation, 309-324
  backward error, 313-316
  Bartels-Stewart method, 311-313
  generalizations, 321–324
  perturbation theory, 318-320
  practical error bounds, 320–321
  solution methods, 311–313
symbolic manipulation package, 6
symmetric indefinite factorization, 218, *see also* diagonal pivoting method
symmetric indefinite matrix, 218
symmetric positive definite matrix, 204
  and block LU factorization, 255–257
  practical test for, 225
symmetric positive semidefinite matrix, 210
  determinantal conditions for, 228
symmetric quasidefinite matrix, 229
synthetic division, 107

tablemaker's dilemma, 5
test

    for accuracy of floating point arithmetic, 54–56
    for guard digit, 56
test matrices, 513–528
    Harwell–Boeing collection, 527
    Test Matrix Toolbox, 591-594
3M method, 450, 461, 505
    error analysis, 458–459
Toeplitz matrix
    pseudospectra, 527, 528 f
    tridiagonal, 524–525
totally nonnegative matrix, 176, 523
    LU factorization, 176, 196
        row scaling in, 192
    test for, 196
transformations, well conditioned, 30
transputer (Inmos), proof of arithmetic's correctness, 59
triangular matrix
    bounds for inverse, 159–161
    condition numbers, 155
    inversion, 265–270
    inversion methods
        blocked, 267–270
        unblocked, 265–267
    M-matrix, 157, 159, 160
triangular systems, 151–168
    accurate solution of, 151 q, 155, 156, 159
    conditioning, 156–157
    fan-in algorithm, 162-164
    partitioned inverse method, 165
    substitution
        backward error analysis, 152–154
        forward error analysis, 155-159
tridiagonal matrix, 183–186
    condition number estimation, 301–303
    growth factor, 183
    LU factorization, 184
        error analysis of, 184–186
    structure of inverse, 303, 305
    Toeplitz, 524-525
truncation error, 6
Turing Award of the ACM, xxvii, 59
Turing programming language, 506
    Numerical Turing, 506
2 x 2 problems, reliable solution of, 500-501

ulp (unit in last place), 43
uncertainty, in data, 5
underdetermined system, 415–423
    backward error, 419, 423
    backward stability, definition, 419
    modified Gram–Schmidt, 421–422
    perturbation theory, 417–419
    Q method (QR factorization), 416
        error analysis, 419-422
    seminormal equations, 417
        error analysis, 421
underflow, 18, 42
    avoiding, 502–504
    effects on software, 504
    model for error analysis, 61
UNICOS library (Cray), 448, 450
unit roundoff, 3, 42
update formula, involving small correction, 30

van der Sluis's theorem, 137
Vancouver Stock Exchange, inaccurate index, 57–58
Vandermonde matrix
    bounds and estimates for condition number, 428
    definition, 426
    inverse, 426–428
    inversion algorithm, 427
    LU factorization in factored form, 432–433
    QR factorization, 441
    structured condition number, 440-441
Vandermonde system, 425-443
    accuracy independent of condition number, 436
    algorithm
        for dual, 431–432
        for primal, 433-434
        for residual of confluent system, 439
    backward error analysis, 437

  - complexity results, 441
  - curing instability, 438440
  - forward error analysis, 435-436
  - history of solution methods, 441
  - preventing instability, 438
- Vandermonde-like matrix
  - confluent, definition, 429
  - definition, 429
  - determinant, 442
- variance
  - algorithms for computing, 12–13, 33
  - condition numbers for, 37
  - error bound for two-pass formula, 38
- vec operator, 310
- vet-permutation matrix, 319, 323
- Venus probe, loss due to program bug, 491 q

- Wedin’s least squares perturbation theorem, 392–393
  - proof, 407–409
- Winograd’s method, 446, 461
  - error analysis, 451452
  - scaling for stability, 452
- wobbling precision, 43, 51
- WY representation of product of Householder matrices, 370-371

- 0°, definition, 64

What is the most accurate way to sum floating point numbers? What are the advantages of IEEE arithmetic? How accurate is Gaussian elimination, and what were the key breakthroughs in the development of error analysis for the method? The answers to these and many related questions are included in *Accuracy and Stability of Numerical Algorithms.*

This book gives a thorough treatment of the behavior of numerical algorithms in finite precision arithmetic. It combines algorithmic derivations, perturbation theory, and rounding error analysis. Software practicalities are emphasized throughout, with particular reference to LAPACK and MATLAB. The best available error bounds, some of them new, are presented in a unified format with a minimum of jargon, and pertubation theory is treated in detail.

Historical perspective and insight are given, with particular reference to the fundamental work of Wilkinson and Turing. The many quotations provide further information in an accessible format.

The book is unique in that algorithmic developments and motivations are given succinctly and implementation details are minimized so that readers can concentrate on accuracy and stability results. Not since Wilkinson's *Rounding Errors in Algebraic Processes* (1963) and *The Algebraic Eigenvalue Problem* (1965) has any volume treated this subject in such depth. A number of topics are treated that are not usually covered in numerical analysis textbooks, including floating point summation, block LU factorization, condition number estimation, the Sylvester equation, powers of matrices, finite precision behavior of stationary iterative methods, Vandermonde systems, and fast matrix multiplication.

Nicholas J. Higham is Professor of Applied Mathematics at the University of Manchester, England. He is the author of more than 40 publications and is a member of the editorial boards of the *SIAM Journal on Matrix Analysis and Applications* and the *IMA Journal of Numerical Analysis*. His book handbook of *Writing for the Mathematical Sciences* was published by SIAM in 1993

For more information about SIAM books, journals, conferences memberships, or activities, contact:

SIAM®
Society for Industrial and Applied Mathematics
3600 University City Science Center
Philadelphia, PA 19104-2688 USA
Telephone: 215-382-9800 / Fax: 215-386-7999
siam@siam.org
http://www.siam.org

ISBN 0-89871-355-2